Spring 2011

A Method to Improve the Sustainment of Systems Based on Probability and Consequences

Michael Ashton Gaintner
Old Dominion University

Recommended Citation

Gaintner, Michael A.. "A Method to Improve the Sustainment of Systems Based on Probability and Consequences" (2011). Doctor of Philosophy (PhD), Dissertation, Engineering Management & Systems Engineering, Old Dominion University, DOI: 10.25777/h1sj-vn35
https://digitalcommons.odu.edu/emse_etds/69

A METHOD TO IMPROVE THE SUSTAINMENT

OF SYSTEMS BASED ON

PROBABILITY AND CONSEQUENCES

by

Michael Ashton Gaintner
B.S. May 2004, Rose-Hulman Institute of Technology
M.S. December 2007, Virginia Polytechnic and State University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ENGINEERING MANAGEMENT

OLD DOMINION UNIVERSITY
May 2011

Approved by:

Shannon R. Bowling (Director)

Rafael E. Landaeta (Member)

C. Ariel Pinto (Member)

V. Dale Bloodgood (Member)

ABSTRACT

A METHOD TO IMPROVE THE SUSTAINMENT OF SYSTEMS BASED ON PROBABILITY AND CONSEQUENCES

Michael Ashton Gaintner
Old Dominion University, 2011
Director: Dr. Shannon R. Bowling

The FROST Method is presented which improves the efficiency of long-term sustainment of hardware systems. The FROST Method makes sustainment and scheduling decisions based on the minimization of the expected value of current and future costs. This differs from current methods which tend to base decisions not on the expected value of costs, but on the expected inventory demand found through projections using data which is often inaccurate.

Distributions are used to account for randomness and inaccuracy in inputs such as failure rates and vendor-claimed dates for end of production. A Monte Carlo technique is then used to convert these distributions into a statistically relevant set of possible futures. Finally, these futures are analyzed to determine what combination of actions will result in the system being sustained for least cost.

Simulations show that, for a realistic range of system parameters, the FROST Method can be expected to reduce the cost of sparing and sustainment engineering between 21.1% and 69.1% depending on the situation, with an average of 43.6%. Implementation involves a slightly increased burden over current methods in terms of the amount of data that must be collected and provided as inputs.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

LIST OF GRAPHS

Graph Page

CHAPTER 1

INTRODUCTION

1.1 The Need for this Method

The sustainment of large deployments of hardware systems has become increasingly complex and expensive. Many of the methods for determining the correct course of action are too simple to deal with this complexity; as a result, suboptimal decisions are made that do not maximize efficiency and minimize cost. At the same time, the increasing costs of sustainment mean that making optimal decisions is more important than ever, as even a small reduction in sustainment costs can be worth billions of dollars in today's environment.

This is illustrated by the current situation of the United States Navy. A consequence of the United States Navy's philosophy of maintaining superiority is that it must maintain a technological edge. This technological edge is becoming more and more expensive (Keller, 2006). It can no longer develop naval systems independently; rather, it must develop them as systems of systems, all of which must be integrated with one another. This puts large additional burdens such as interoperability and information assurance on the designers of these systems. In addition, economic realities have caused the Navy, which was once upon a time the driving force behind many technological developments in electronics, to be at the mercy of the commercial industry's development schedule.

Together, these factors are causing Navy costs to skyrocket. The new ship classes and systems being developed have price tags that are all but unaffordable. As a result, the Navy cannot afford to build its way out of maintenance problems; instead, it must sustain its current fleet while adding new ships when possible. This has caused the Naval Sea Systems Command to identify the need to "sustain today's fleet efficiently and effectively" as one of its three core goals in its Strategic Business Plan (Naval Sea Systems Command, 2009). This sustainment currently costs the Navy approximately $8 billion per year.

While the US Navy example is a striking one heavily referenced throughout this paper, it is not alone. Corporations, governments, and universities are now expected to provide their employees with computers and dedicated intranets, with the largest ones such as for the Department of Defense costing billions of dollars (Shachtman, 2010). Similarly, hospitals must constantly deploy the latest in equipment and information management if they want to remain competitive, and that means billions of dollars spent on technology (Center for Environmental Health, 2010).

For many, the sustainment and scheduling of electronic systems is a small enough issue to be managed simply, and good efficiency is good enough. For the more massive deployments starting to be encountered, however, another level of effort is called for. Investing the effort and money to eke out even small improvements in efficiency can result in billions of dollars saved over the lifetime of some of these systems due to their extreme costs.

The goal of this work is to trim these costs by developing a method to move towards optimized decisions in many aspects of the sustainment of electronic systems. Ideally, this improvement will eventually come about through the development of software which will provide decision makers with previously unavailable information in order to make the best possible decision in various scenarios involving electronics sustainment. This dissertation will present the theoretical method upon which this software and process will be built.

1.2 How Better Choices Can Be Made

Imagine you are responsible for sustaining all deployments of a large, multi-million dollar electronics-based system. There are dozens of copies of this system already deployed, and a few more copies that will be deployed in the future. An electronics-based part in this system, such as a processor or network card, just failed. After replacing it you see your inventory of spares for this part is getting low. You call up that part's vendor to order a few more and are told that they are going to stop production in a few months. This is your last chance to purchase all the parts you are going to need for the life of that system. How many should you get?

This is a normal situation for many engineers and logisticians to find themselves in. A typical response would be to check records and find, as an example, that inventory has dropped by 20 over the last four years, indicating a rate of about five per year. Someone who knew they had to support these systems for 10 more years could extrapolate that, at five parts a year, they will need 50 parts. At this point an understandable action might be to add a few more parts just to be safe and go execute an order for 55 parts.

This reactive method is fairly common these days, and scenarios just like it occur regularly. This simple method is surprisingly effective and can, in fact, sustain systems. It is not, however, efficient. 55 is probably not the ideal number of parts to have purchased. Additionally, it was only through luck that you happened to find out the part was going out of production. Without this luck, you might not have noticed that the part was going out of production until it was too late to procure additional spares.

Luckily there are better, more proactive methods available. Imagine the same scenario above, but now a modern obsolescence management process is being used. This time, thanks to sharing of information through contacts and databases, the process alerts logisticians a full six months ahead of time that the vendor is stopping production. They have plenty of time to determine how many parts to purchase, but they do not even need that time because the process dictates how many to buy. Even better, the process employed is smart enough to consider the fact that there are a few systems that are yet to be deployed; as a result, logisticians learn that once those additional systems come online they can expect to see an increase in failures. Thanks to this consideration, the process gives better information than is available using the reactive method. In the same situation where you earlier bought 55 parts, the more proactive process might look a little deeper and find that while the system may be experiencing five failures a year for now, that can be expected to increase to seven failures a year once the new systems come online. The process determines that 60 parts should be ordered, and they are.

This seems to be a far better situation to be in. The risk of missing an opportunity to make a last purchase is reduced to almost zero, the process runs smoothly, and everything seems to be working just fine. Most anyone who sees the system in place would be impressed and walk away confident that the job is being done well.

This is the state of the art method being used in many situations today, and it is, in fact, effective. Given the limited amount of data that has been presented so far, it even seems efficient, but upon further examination, it is not. If viewed with additional information, it can be shown that the solutions above were poor, and there is significant room for improvement.

Imagine the same scenario yet again, but this time more information is available. This time around, an engineer responsible for this system realizes that the predictions of the number of failures she should expect are not perfect. 60 parts may be the average number of parts needed, but it is likely that the real number will end up being slightly higher or lower. Knowing this variance exists, she decides to come up with a simple equation that relates the quantity of spare parts in inventory versus the likelihood of that inventory being sufficient. She knows that if she has zero spare parts in inventory, she should have zero confidence that the system is covered should a failure occur. She also knows that having 100% confidence would require an infinite number of parts. Through data mining, she manages to come up with a relationship which matches the previous process' result that 60 spares will give a 50% probability of not running out of inventory. For this particular part, that relationship turns out to be

$$X = \frac{118P^2 - 119P}{P - 1} \tag{1.1}$$

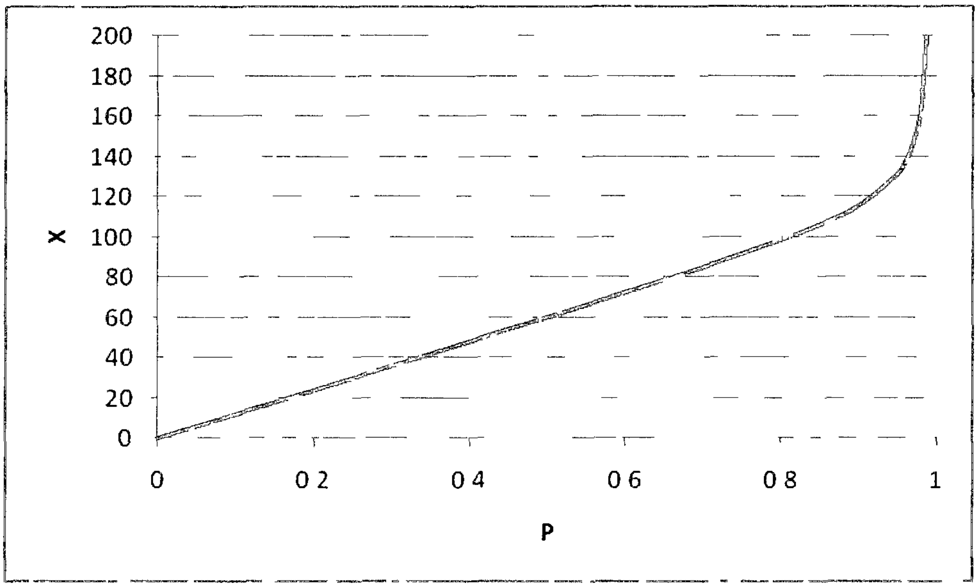


Figure 1 - Probabılıty of Sustaınıng System vs Number of Parts Purchased

where X is the number of parts needed, and P is the probability of not running out of parts. Now that she has a relationship between how many parts to buy versus how likely she ıs to be able to support the system in the future, she wants to use this relationship to sustain the system as cheaply as possible. She decides to start by determining how much it would cost to perform the solutions the other methods already came up with. Then, she will find out how she can improve on that.

Before determining how much these solutions cost, a little more data is required. Assume that this part is a key processor which the system cannot function without. The system uses processor-specific code, and if inventory runs out and engineers have to switch to a different processor, it will cost \$2 million to rewrite that code. Also assume that each processor ordered costs \$1,000.

There is now enough information to determine the expected cost for the solution that the proactive method recommended. It said the engineer should purchase 60 parts at \$1,000 each for a total cost of \$60k. Since this was based on average failures, this will leave a 50% chance of running out of parts and needing a \$2M redesign. Thus, the expected cost is:

$$\text{Expected Cost (proactive)} = \$60\text{k} + 50\% \times \$2\text{M} = \$1{,}060{,}000. \qquad (1.2)$$

But what about the earlier scenario where the proactive method was not used? Under the original, reactive method, the engineer was going to purchase 55 parts. Using the equation above, a purchase of 55 parts would give a 54.11% chance of needing a redesign.

$$\text{Expected Cost (reactive)} = \$55\text{k} + 54.11\% \times \$2\text{M} = \$1{,}137{,}171. \qquad (1.3)$$

Clearly the proactive method did a better job, saving $77k on this one part alone, but the point is to sustain this system as effectively and efficiently as possible. How would the expected cost be minimized? The equation for this cost is

$$\text{Expected Cost} = X * \$1k + (1-P) * \$2M \tag{1.4}$$

Plugging in the previous equation for X, the cost equation becomes

$$Expected\ Cost = \$1k\frac{118P^2-119P}{P-1} + \$2M(1-P) \tag{1.5}$$

Minimizing this gives a result of P=97.71% where X = 158. In other words, the optimal action in this situation is to purchase 158 parts, reducing the chance of a major redesign to only 2.29%. This lowers the Expected Cost to $203,766, less than 18% of the cost of the first attempt.

The first two methods presented, reactive and proactive, were based entirely off of need. If it seemed like 60 parts were needed, 60 parts were acquired. This new method added another consideration: what will the consequences be if inventory runs out? In this example, it turns out that the consequences were pretty severe. In order to reduce the chance of having to face that severe consequence, the right solution was to purchase more parts than we will probably end up requiring.

This demonstrates the problem with current best-practices. They simply are not considering all that needs to be considered to come up with the correct solutions. This example was chosen to dramatically make the point, showing how more than 82% of the cost could be avoided by considering consequences and probabilities. The reality is generally not as bad as this example. For important systems, software rarely calculates the 50% point since running out of inventory for 50% of the parts in a system would almost never be considered acceptable. The typical solution is to pick a "rule of thumb" which achieves good results. One possible simplification is to be conservative and go with 95% probability regardless of situation. This comes much closer. In fact, in this scenario the conservative 95% choice would reduce the expected cost to $231,387. That is still 13.6% more expensive than the optimal solution, but it is far better than the previous attempts.

Method	Parts Purchased	Odds of Redesign	Expected Cost
Simple (reactive)	55	54.11%	$1,137,171
Proactive, average estimate	60	50.00%	$1,060,000
Proactive, conservative estimate	131	5.02%	$231,387
Optimal	158	2.29%	$203,766

Table 1 - Expected Cost for Methods in Example 1

Table 1 shows the four different solutions discussed for this example. It quite clearly shows a dramatic improvement between the first two methods and the optimal one. However, it also shows the conservative estimate, at 95% confidence, performing almost as well as the optimal method. At this point one might think that this 95% conservative method is the way to go; it performs almost as well as optimal but it requires quite a bit less effort. Before proceeding, one more example will be provided which dispels that idea.

While it worked just fine in the first example, it turns out that simply assuming that a high probability is always the best option is a horribly inefficient method as well. Imagine the exact same scenario and equation, except now instead of a key processor the part in question is a ruggedized DVD-ROM drive. It still costs $1,000 to purchase one, but there is a difference in the consequence of running out of inventory. In this case, an engineer could simply choose a new drive from another vendor. Substituting a new drive would require $10k worth of testing to ensure the new drive works just fine in the system. Substituting the drive, however, does not magically mean you no longer have to purchase drives; in fact, the new drives cost more, at $2k. Thus, if too few of the current drives are purchased upfront, we will have to pay double for each additional drive later plus a $10k testing fee. For simplification purposes, assume that if inventory runs out, the final amount needed is at the halfway point of the remaining probability. For example, if we purchase enough drives to have a 50% chance of not needing a substitution and we lose that bet, we will need enough of the new drives to bring us from the 50% point to the 75% point.

Expected Cost = Old Drives Purchased + Testing Fee + New Drives Purchased

$$Expected\ Cost = \$1k \cdot X[P] \ + \ \$10k \cdot (1 - P) \ + \ \$2k \cdot \left(\frac{X[P-1]}{2} - X[P]\right) \tag{1.6}$$

$$Expected\ Cost = \$1k \cdot \frac{(117P-120)}{P-1} \ + \ \$10k \cdot (1 - P) \tag{1.7}$$

Minimizing this cost gives an ideal scenario where 54 parts would be purchased up front. This would lead to a 45.07% chance of needing no further action, and a 54.93% chance of needing to qualify the new drive and purchase additional parts, for a total Expected Cost of $127,955.

In comparison, the "conservative" method, which is often used, would instead come up with a cost of $177,263.

Method	Parts Purchased	Odds of New Drive	Expected Cost
Simple (reactive)	55	54.11%	$127,955
Proactive, average estimate	60	50.00%	$128,000
Proactive, conservative estimate	131	5.02%	$177,263
Optimal	54	54.93%	$127,955

Table 2 - Expected Cost for Methods in Example 2

In this situation, the results of the methods are reversed from the last example. The reactive estimate, which performed terribly in the last example, actually came within $1 of the optimal solution this time. On the other hand, the proactive, conservative method performed the worst, costing almost 39% more than necessary.

These examples illustrate that any "rule of thumb" method used to determine sustainment solutions cannot hope to achieve anything near ideal efficiency. The ideal confidence is situation-dependent; in the first example the ideal solution involved paying enough to get almost 98% confidence that the system could be supported with no further action. In the second example, the ideal solution involved only paying enough to get 45% confidence. This shows that you cannot hope to achieve ideal efficiency by using any simple rule. To reclaim this lost efficiency, a method is needed which enables the sorts of calculations demonstrated above. This method must be able to determine the probability of a solution succeeding and the consequences if it does not.

If such a method existed, it could be used to determine the ideal least-cost solution for a scenario. Additionally, multiple scenarios could be tested using this method to see which scenario results in the least expensive future. Defining a way to achieve this new ability is the goal of this paper. This dissertation creates a probability-based method which improves efficiency by enabling consequence-based decisions and also allows someone to compare scenarios and determine which one is likely to be the least expensive in terms of sustainment costs.

The proposed method finds ideal solutions in order to provide long-term supportability of hardware systems and ensures continued functionality, which is part of what the US Navy refers to as "Future Readiness." This first real-world implementation of this method will be the creation of a Navy tool called the Future Readiness and Optimized Solution Tool, so through the remainder of this paper the proposed method will be referred to as the FROST Method.

1.3 Outline

The remainder of this paper is broken into nine key sections. The first section contains the initial steps of a Systems Analysis. This analysis frames the problems being tackled at a high level and attempts to set the path for the more detailed aspects of the research while minimizing the likelihood of running into errors. In this analysis, the problem situation is presented along with a demonstration of how improving this situation will be useful and valuable. Additionally, a basic methodology called the FROST Method is provided to improve this situation, the method is shown to be new, and a demonstration is provided to show how it will be validated.

The next section provides the detail design and theory. This discusses in depth the mathematical model and distributions which comprise the method. Additionally, an equation is developed which can be used to estimate the reduction in cost the FROST Method will provide over current methods. The remaining sections will be focused around finding the values to evaluate this equation.

The third section provides a discussion of the distributions found through data mining and how they can be used to include randomness in predictions.

The fourth section evaluates the FROST Method's ability to correctly choose a better schedule and compares it to current methods.

The fifth section explains how the data provided by the FROST Method can be used to choose more optimal solutions and model their costs.

The sixth section demonstrates that the results in sections four and five are independent and can therefore be readily combined.

The seventh section combines all of the results and quantifies the improvement over current methods.

The eighth section provides a basic framework that can be used to adapt the FROST Method for use with non-electronic parts and systems.

Finally, the ninth section concludes.

CHAPTER 2

ANALYSIS

2.1 Initial Framing

2.1.1 Background

Before attacking the problem, it is important to understand the background which has caused this situation. For this background, the focus will be on the Navy scenario presented in the introduction. While only the one scenario will be presented as an example, it is worth noting that the Navy is not alone. The organizations involved in the previously mentioned intranet and hospital scenarios also rely on commercial electronics and face similar scheduling and harvesting decisions, as do many others. The Navy is just one example of a situation where this methodology could be applied successfully.

Early in the history of electronics the United States Navy and the Department of Defense were driving forces behind much of the electronics world, and the electronics industry moved at a pace the government could keep up with. Much of the movement in the electronics industry actually came from government funding and research, giving the government a large amount of control over the electronics world. This significant degree of control over industry made things relatively simple for the government in terms of system supportability. If the Navy wanted an electronics system aboard a ship for a significant number of years, it had the weight to affordably contract the necessary electronics to be created and supported for the life of the system.

This is no longer the case. In the 1970s, the military purchased 35% of domestic semiconductor production. By 1984 this had dropped to 7%, and by 2001 the military represented less than 1% of the market (Hamilton & Chin, 2001). Defense spending is now eclipsed by the rest of the electronics market. The US military cannot afford to keep up with industry on its own. It faced a choice to fall behind or adapt by leveraging the advances of the commercial market, and it chose to adapt. In 1994 Secretary of Defense William Perry issued a memorandum declaring that the DoD would use Commercial-Off-The-Shelf (COTS) technology whenever possible (Perry, 1994). As a result, entire systems are now being created using COTS technology and this is significantly shifting the burden of system life-cycle engineering. Design has been simplified as a large portion is now left up to industry. Sustainment, on the other hand, has been made far more difficult as control over much of the process had to be forfeited. For example, a commercial processor will likely only be produced and procurable for a year or two when the Navy would like to use it aboard a ship for over a decade. This is a sizeable problem for the future of the Navy, as electronics costs are growing and represent a large portion of the budget (Keller, 2006).

Since these changes and the forfeiture of control over many of the variables impacting sustainment, the Navy has not managed to put in place a complete process that can efficiently support the electronic systems that are the heart of the modern navy. There are many individual solutions in place to improve small aspects of the sustainment problem, but there is nothing that comes full circle and allows the sorts of feedback loops that would cause the Navy to truly capture the efficiency they could be realizing.

The reason is that the systems and variables involved are incredibly complex. Wrapping one's mind around the problem as a whole is all but impossible, so engineers and logisticians have done what they do best: they have broken off pieces small enough that they can understand them and tried to solve those pieces. These simplifications allow for sub-optimization by making the smaller problems simpler to solve, but they make it difficult to consider the big picture.

For example, consider two separate tasks in the lifecycle and management of a major electronics system. First, high-level management is making many fielding schedule changes that alter the dates when electronic systems are fielded. This could be the Navy deciding to delay the upgrade of a combat system aboard a ship until the next time it comes to port, or it could be a company pushing up the date a new data center is deployed because of increasing demand by customers. These sorts of changes happen frequently due to many types of issues, whether political or financial. Second, logisticians are trying to keep apprised of inventory needs to ensure they always have spares available for these electronics systems. These two problems clearly affect one another. If a system is going to be fielded two years later than planned, that will have an effect on the purchases the logisticians need to make to maintain the inventory required to support the system. If these new purchases are going to be expensive, knowledge of that additional cost might alter the decision to make the fielding schedule change in the first place. These two tasks are very clearly related and should be considered together to avoid sub-optimization, but in reality, the current state is that logisticians have created simplified models that get the job done from their perspective, often without even considering schedules as a variable. Current methods do not consider consequences, and often they do not even consider costs or schedules. As an example, official Navy policy shows two main types of projections that are used to cover the majority of ship and shore electronics based systems (Chief of Naval Operations, 1999). These include stocking inventory to a "Fixed Protection Level" which "computes allowances on the basis of a single factor, demand" and a "Variable Protection Level" which "computes allowances on the basis of several factors, e.g., demand, item price, and item essentiality." The Navy spends $8 Billion a year sustaining systems (Naval Sea Systems Command, 2009) yet does not consider the financial consequences of individual sustainment decisions. At worst, it uses a Fixed Protection method which can mean the simple demand consideration of seeing that 10 processors failed last year and assuming that 10 will fail next year, regardless of any information about the schedule. At best, it uses a Variable Protection method which attempts to stock more parts which are "essential" or cheap and fewer parts which are less essential or expensive.

The result of this lack of consideration of consequences as well as often a lack of consideration of schedule is that managers may unknowingly make a tremendously expensive schedule change, and then logisticians, unprepared for the effects of this change, may run into problems. If this situation could be resolved so that these groups are aware of their effects on one another, more efficient decisions could be made.

2.1.2 Framing Method

Before developing a method to improve this situation, the problem needs to be framed. The first step in framing the problem is to identify a set of applicable system principles which can likely be exploited. The second step is to convert these exploitable principles into problem statements which identify areas to attack. Next, an evaluation is performed to find how the situation's context is likely to impact the ability to make improvements. Last, potential design issues are predicted to decrease the probability of running into any errors. This should complete an initial framework and pave the way for the development of a methodology to improve the situation.

2.1.3 Exploitation of System Principles

Since COTS parts have led to a challenge in efficiently supporting electronic systems, any proposed solutions must be able to demonstrate that they are in fact able to support the systems in question. For my purposes I will very early on make an assumption using the Redundancy of Resources Principle. This principle basically says that stability can be improved by ensuring the redundancy of important resources (Clemson, 1991). In this case, I will assume that a system can be considered as supported to the point of stability if at all times there are spares available for any important pieces of electronics. In other words, any acceptable solution will have to meet the requirement of maintaining positive inventory at all times for all important electronics, and the goal is to do this as efficiently as possible. This assumption will later be discussed in more detail.

The second principle to consider is the Sub-Optimization Principle. This principle says that the optimization of a subsystem may actually worsen the performance of the system as a whole (Skyttner, 2001). For example, as presented above, choosing the most convenient fielding schedule may actually decrease overall system supportability. This is where many of the chances to make improvements in efficiency will come from. The goal is to move away from sub-optimization and towards real optimization. For this to occur, these subsystems need to be able to exert some level of control over one another instead of acting as completely independent subsystems.

This can be accomplished through consideration of a third systems principle, the Conant-Ashby Theorem. If these subsystems are to have some degree of control over each other, there needs to be a regulator to regulate this control. The Conant-Ashby Theorem says that "every good regulator of a system must be a model of that system" (Clemson, 1991). Therefore, in order to get away from sub-optimization, there needs to be a system model which can serve as a regulator.

This brings up a fourth systems principle, the Requisite Variety Law. This law says that the ability of a regulator to do its job is going to be dependent on its variety and its communication channel with the system it is trying to regulate (Ashby, 1958). In other words, the model that needs to be developed must sufficiently consider all important variables, and it must have sufficient communications with the system it is trying to control.

This leads to a fifth systems principle which requires leaping ahead to system context which will be discussed in more detail later. The reality is that these subsystems which the model needs to regulate are not subsystems that the model can realistically expect to have strong, direct control over. In the naval example, these subsystems include people ranging from engineers and logisticians to Admirals, members of the Senior Executive Service, and Congress, who between them make decisions on budgets and schedules which play a major role in this overall system. Likewise, the players in the non-naval examples who are making decisions on whether and when to field systems are going to be senior members of their organizations. It is not realistic to directly control these players. Instead, the Self-Organizing Systems Principle must come into play. This principle says that a complex system will organize itself and its behavior will be the result of interaction amongst the subsystems (Clemson, 1991). In other words, the idea is not to directly force a change in behavior of these players but rather to provide the right interaction amongst the systems they control so that better methods become apparent and assume they will adjust their own behavior to take advantage of these opportunities.

2.1.4 Problem Statements

When combined, these systems principles lead to three problem statements and associated perspectives.

1. Positive inventory of electronics needs to be maintained in an efficient manner for these systems.

2. A model needs to be developed that can serve to regulate the processes that affect the supportability of systems and remove inefficiency.
3. Opportunities to improve sustainment efficiency need to be made apparent to those able to reorganize systems and processes.

2.1.5 Context, Perspective, and Design Issues

The first problem statement is the most straightforward as it can be turned directly into a fairly unambiguous high-level requirement. At lower levels, however, there is context that could lead to design issues.

First, what qualifies a part as important enough to mandate a positive inventory at all times, and warrants the amount of effort that will likely be required to make a piece of electronics part of the model? The Bill of Materials (BOM) for many systems can be thousands of parts, many of which are screws, fasteners, indicator lights, etc. While it is true it would be good to have these spared, the level of effort required to track inventory and failure rates, as well as make database entries, etc. in order to optimize the number of screws kept on hand is beyond what could reasonably be asked.

At the other end of the spectrum, there could be electronics that are so expensive that they do not necessarily warrant sparing. If there is an overly expensive part that is not certain to ever need replacing, it is still being produced, and having a delay in replacing it is acceptable, then the optimal situation for this part might be to actually have zero spares on hand and order replacements only when needed. This would be an exception to the problem statement, and the possibility needs to be allowed in any developed model.

The second problem statement adds a modeling perspective whose context brings up several design issues. First, as the goal of the model is to regulate the subsystems involved such as engineering, purchasing, scheduling, etc., the model is going to need to include all of these aspects. However, it needs to do it in a reasonable manner; the end goal is improved efficiency, so this model cannot overly burden its users with requirements or else any gains in efficiency will be traded away with the need to support the model.

In addition, this model is going to have to be dependent on inaccurate data. Modeling future situations is going to require using estimates of the future as inputs. These estimates are going to often be wrong by their very nature, so the model cannot simply assume all of its input data is correct without making bad predictions. Uncertainty in input data means there will have to be uncertainty in the results, and this should somehow be statistically quantified in a way that gives users of the model some idea of the likely accuracy in the results. This uncertainty in the future, once quantified, should be able to be used to provide the probability of running into future consequences, as discussed in the introduction.

The third problem statement involves incorporating people into the model, and this is a concern because the behavior of people is rarely easy to predict with significant accuracy. The real-world perspectives of these people need to be considered in order to determine exactly how they should be incorporated.

For example, in order to be able to compare costs and make optimal decisions, one of the costs that will need to be considered is engineering costs. To a large degree these can probably be automated in the model; given two engineering solutions that are truly equally valid, it is relatively safe to model engineers as inherently choosing the most cost-efficient solution. If the model can determine the cheapest solution, it can go ahead and assume the solution will be applied by engineers reading the model and move forward with its predictions.

On the other hand, scheduling decisions made by people are another key consideration. Given two scheduling options, a good model would also be able to determine which schedule is the most cost efficient. However, there are political considerations to scheduling beyond cost. In the Navy example, delaying the fielding of a system could potentially alter the behavior of another nation or even affect the outcome of a battle. In commercial situations, changing a delivery schedule could impact a company's reputation or their market timing. Making the model choose the most cost-efficient schedule and move forward with predictions based on sustainment cost alone would be a mistake, as it is possible, or even likely, that the most cost-efficient schedule will not be the one actually used. This needs to be taken into consideration when determining how to handle such situations with the model. In this case, this will be considered by using the standard policy analysis technique of simply generating and presenting objective information to policy decision makers and leaving it up to them to apply the information and make decisions with it (Thissen & Twaalfhoven, 2001) (Miser & Quade, 1985).

Now having a set of problem statements, a contextual analysis, and design issues to avoid, it is time to develop a methodology to improve the situation while taking these aspects into consideration.

2.2 Methodology

2.2.1 High-Level Methodology

It has already been determined in the problem statements that a model is required. The model will need to accept realistically available or obtainable data as an input and provide an objective, measurable output. Additionally, the model will need to maintain positive inventory and be capable of dealing with uncertainty. At the highest level, a simple diagram of the model is shown in Figure 2.

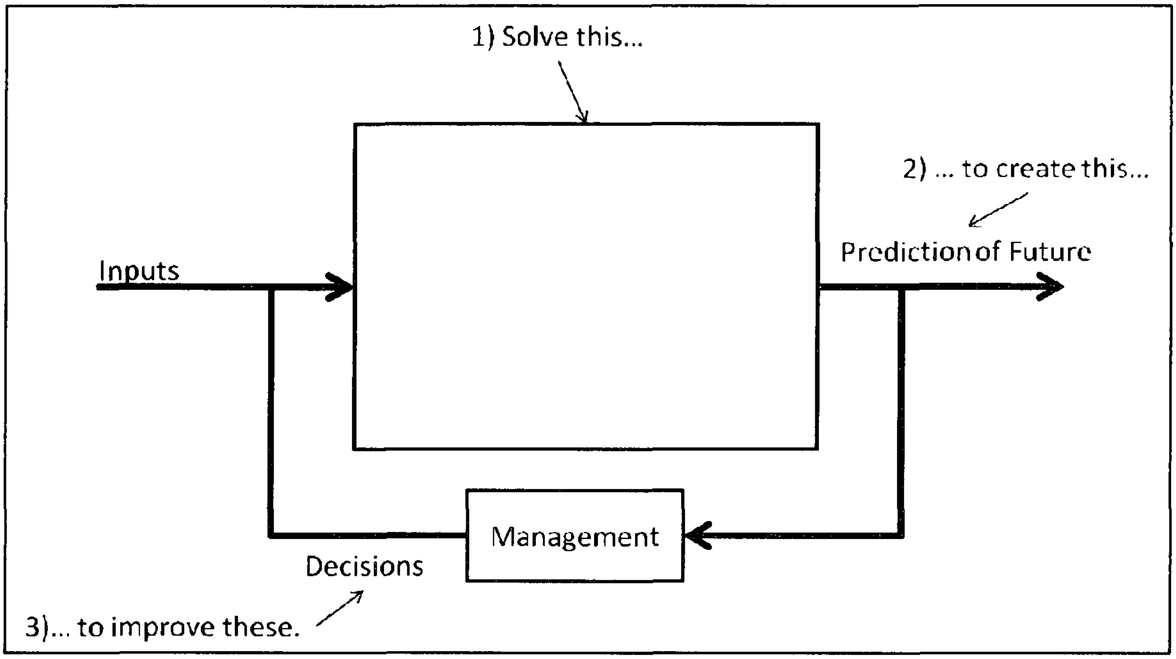


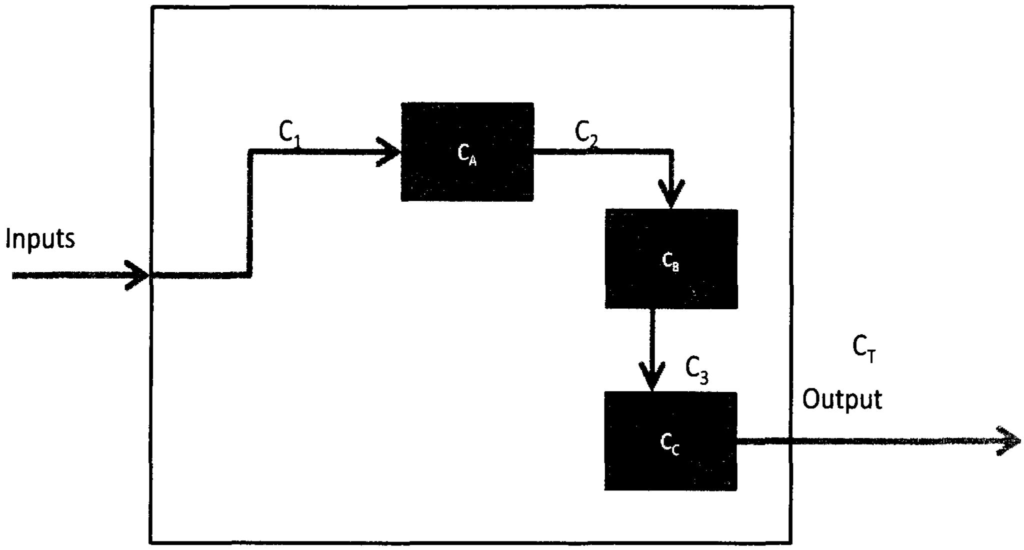
Figure 2 - High-Level Method

2.2.2 Validation Method

Before drilling down into a more detailed methodology, the validation method needs to be discussed. With this sort of model, the obvious path forward is to validate it empirically by demonstrating a strong correlation between the output of the FROST Method and real-world data. Unfortunately, sufficient real-world data will not exist until well after the timeframe of this dissertation. As a result, the method will have to be validated rationally. There is an inherent weakness in validating a complex model rationally, and this needs to be addressed before presenting the method.

Virtually all complex engineering and modeling has uncertainty associated with it. We have variables like error rate, correlation, confidence, and probability to address this fact. When validating a model, regardless of the method used, what is really being demonstrated is that there is a high enough confidence in the output and its relationship to the real world to meet the intended purpose. For example, it may be determined that there is a threshold of 90% confidence (or correlation with the real world) in order for a particular model to be considered valid and the theory behind it to be knowledge. If the confidence can be shown to be above that threshold, it is valid. There is a tremendous amount of debate around this subject and many ways to think about the concept of knowledge, but it all boils down to this concept. If an idea is shown to have strong correlation to reality and the idea makes enough logical sense that it can be believed there is causality behind this correlation, there is justified true belief and therefore knowledge. This is a very important concept in a paper that is required to make a contribution to knowledge.

Complex engineering and models involve many parts or modules, and each one of these can be a source of uncertainty and error. Each additional step may introduce error and further remove the output from perfectly correlating with reality. In addition, the linkages between modules can also be a source of uncertainty. If these sources of uncertainty are independent and linear, they chain together similarly to the way we consider reliability models. See Figure 3 as an example.



$$C_1 \times C_2 \times C_3 \times C_A \times C_B \times C_C = C_T = 0.96$$

Figure 3 - Independent Errors Chaining

There are two downsides to using a rational method instead of an empirical method to validate a complex model. The first downside is that confidence in rational knowledge does not necessarily have an exact value associated with it. Even the uncertainty can be uncertain. For example, most people could watch a major league baseball game and rationally come to a conclusion that they would be unable to hit a fastball from a major league pitcher. They might feel they have sufficient confidence to meet their threshold of certainty without knowing an actual value for that confidence. Would they be successful one out of ten pitches? One out of one thousand? For many, discovering an exact value is not necessary to form a belief based on rational certainty. As a result, rational certainty is not always quantified, which can cause a difficulty that will be demonstrated in an example below.

The second downside is that rational methods require considering all of the terms in the example equation in Figure 3 above. When validating empirically, there is no need to consider the left-hand terms. C_T can be measured directly to determine if it meets the threshold. Rationally, however, the only way to have any idea of the magnitude of C_T is to estimate or calculate it by analyzing every module and link that affects C_T, which is what makes up the left-hand terms.

When combined, these downsides can cause a problem for complex models. Assume there is a complex model with 20 independent components, each component has been individually rationally validated, and they have been linked together in a logical manner. Because they have been rationally validated, it is known that the confidence in each component is high but there is no defined confidence. The question is, when combined, do these 20 components still result in a valid whole? There is no way to know. This can be shown through math. If there is a 99.5% confidence in each of the twenty components, the total confidence is 0.995^{20} = 90.4% confidence, which is probably high enough to be valid. On the other hand, if there is a 95% confidence in each component, the total confidence is 0.95^{20} = 35.8% confidence, which is probably not high enough to be valid. So, given a complex combination of many rationally validated components, the best that can be said is it is *possibly* valid.

This issue potentially extends to this dissertation. There will be four modules presented with three links connecting them. It will be shown that each module and link individually can be viewed with some reasonable degree of confidence. Logical arguments will be formed that each module individually “makes sense” and feeds into the next, but without being able to make an empirical correlation between the output and the real world, how can it be demonstrated that the resulting combination still has sufficient confidence to be valid?

This will be accomplished in four parts. First, each module and link in the model will be individually validated and shown to have reason behind it. Second, the model will be scoped and the assumptions defined in such a way to minimize sources of imperfect correlation, resulting in three primary sources of imperfect correlation. This will serve both to eliminate sources of low confidence and to keep the scope of the dissertation realistic and manageable. Third, it will be demonstrated that these sources of imperfect correlation are independent and can therefore be simply combined using math. Fourth, simulations will be run which incorporate these three sources of imperfect correlation. Since the imperfect correlation involved in these simulations will be shown to be independent, the results of these simulations can then be combined. This will allow the actual performance of the method to be quantified while including these sources of imperfection. This negates the requirement to quantify the output's correlation with reality by providing an alternate threshold. Instead, the threshold will be a comparison of the performance of the FROST Method against current methods. It will be demonstrated that the FROST Method manages to provide significantly improved results over currently available methods. As a result, although the correlation will not be quantified, it will be included in the results and shown not to hinder the method's ability to provide improved performance.

The problem being solved makes it an impossibility to provide a generic, situation-independent performance measure. As a result, the performance demonstrated will be situational. Examples will be provided showing the expected performance under a wide variety of different situations. Consider the graphs in Figure 4 and Figure 5, each showing cost predictions for two different scenarios.

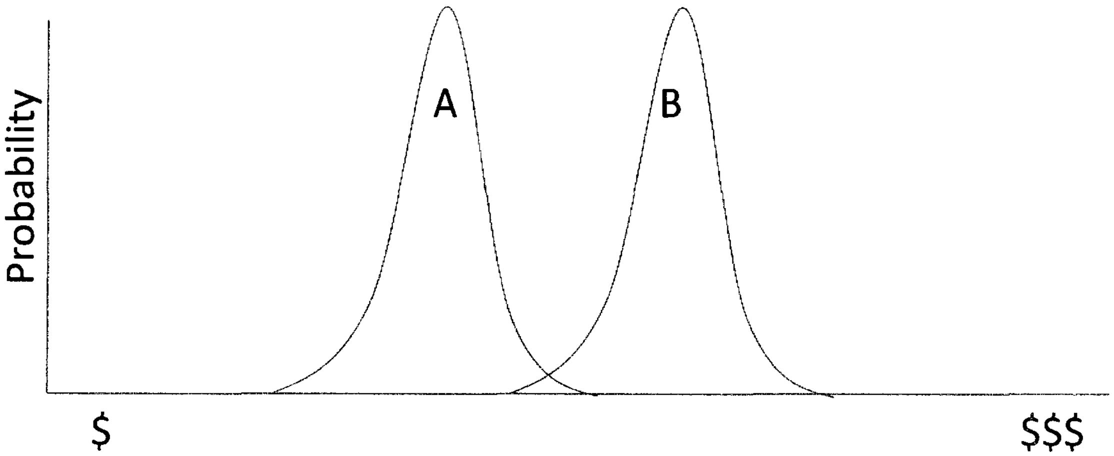


Figure 4 - Clear Difference between Scenario A and B

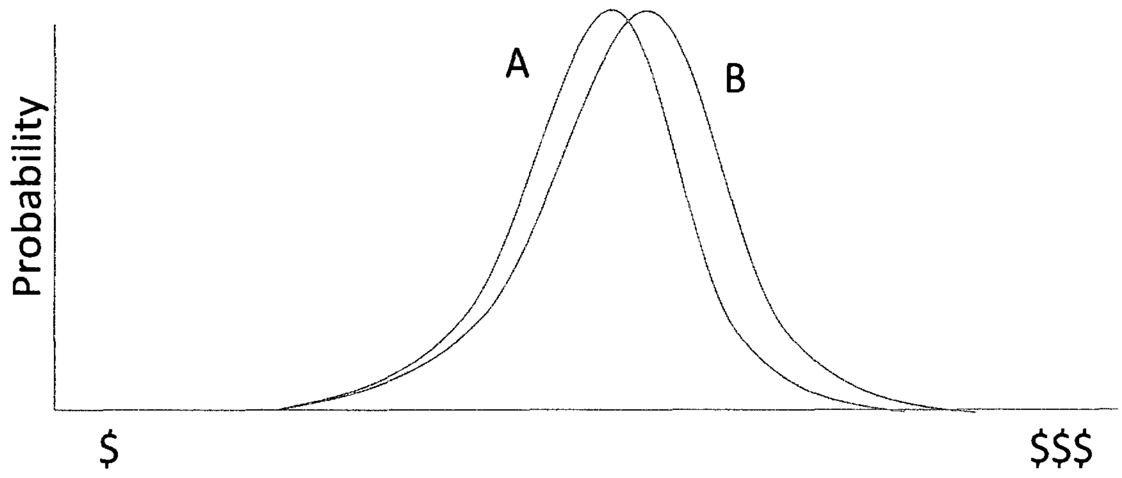


Figure 5 - Slight Difference between Scenario A and B

The situation in Figure 4 is clearly preferred. Each scenario is very confident that its cost will be in a small range; as a result, it can be said with high certainty that option A is better than option B. The graph in Figure 5, on the other hand, shows a scenario with less certainty. In this situation, one would still make the choice of A over B, but it would be far from certain. This would still be an acceptable, successful model, even with low confidence. However, the performance could be expected to be worse in such a situation compared to the previous example. This can be demonstrated with a mathematical example. Consider a model with probability P that choice A is better than choice B. Each correct choice saves $X, and each incorrect choice costs $X. N choices are made. The resulting expected value of savings would be:

Confidence x Savings + (1-Confidence) x Loss =

$$\sum_{i=1}^{N}(P)(X)+\sum_{i=1}^{N}(1-P)(-X)=(2P-1)NX \tag{2.1}$$

Assume this model is applied to 20 systems and each system uses it to make 10 decisions a year with an average potential savings/cost of $100k. This means a savings of (2P-1) * $20M/yr. Even for a difficult situation that only results in a 51% confidence that choice A is better than choice B, this still would represent an expected savings of $10 Million over 25 years. On the other hand, a better situation that resulted in a 90% confidence would provide an expected savings of $400 Million over the same time period. While it is evident that both are excellent results, there is a dramatic difference in performance in these different scenarios, which is why the performance will need to be demonstrated situationally.

As one last consideration, the long-term performance of this method can be expected to increase over what is shown by building feedback into the method that can utilize empirical data when it eventually becomes available. This will essentially allow the model to tune itself based on observed data, meaning that whatever confidence it manages to originally start with can be expected to rise over time.

In summary, the method will be scoped to include three main sources of imperfect correlation, and it will be demonstrated that they are independent and combinable. The different modules of the method will be simulated while including these sources of imperfect correlation. The results of these simulations will then be combined in order to show that they still result in performance beyond what is currently available. Additionally, feedback mechanisms can be built in so that when empirical data becomes available it can be used to further improve the model's correlation and performance.

With these steps complete, it will be evident that the FROST Method delivers improved performance and that it can be expected to further improve over time. This will sufficiently demonstrate that the method is useful, viable, and valid.

2.2.3 Detailed Methodology

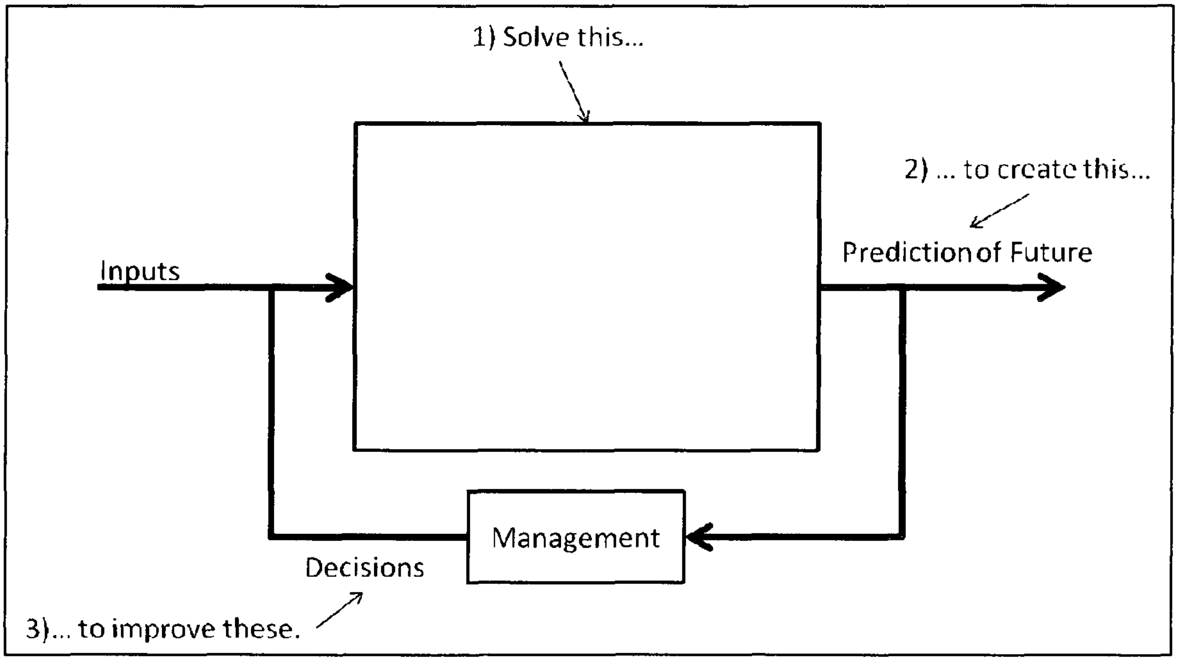


Figure 6- High-Level Methodology

As shown previously, what is desired is a method to create the feedback mechanism shown in Figure 6. If this box can provide feedback to management on the effects their decision will have on the future, they can test out their decision beforehand and use this information to make more optimal choices.

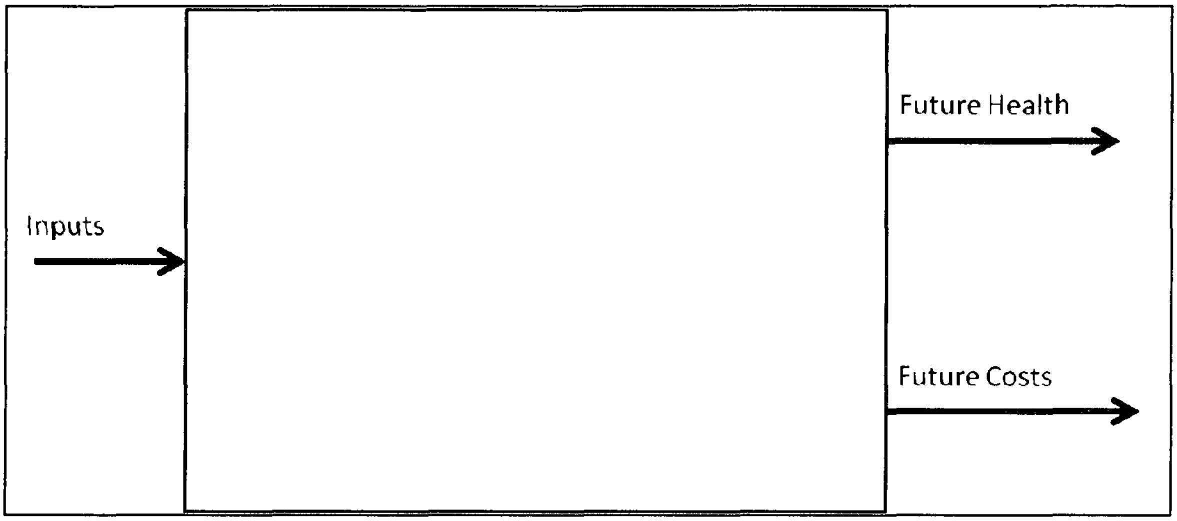


Figure 7 - Sustainment Decisions Center Around Health and Costs

The "Prediction of Future" shown as the output in the high-level methodology is, so far, vague. The management of engineering projects is typically concerned with the three major factors of cost, schedule, and performance, so these need to be reflected in the model. Since this model is for the sustainment phase of a system, Schedule and Performance take on slightly different meanings. There are not delivery dates and performance specs that must be met; instead, what matters is that the system remains healthy (performance) throughout its required life-cycle (schedule). In other words, the relevant factors are Future Costs and Future Health as shown in Figure 7.

These are the key factors for this model. If the cost is predicted to be extremely high, that feedback is very likely to cause management to look at decisions which could lower the cost. Similarly, if system health is predicted to be unacceptably low, the feedback loop would cause management to look for decisions that could improve health.

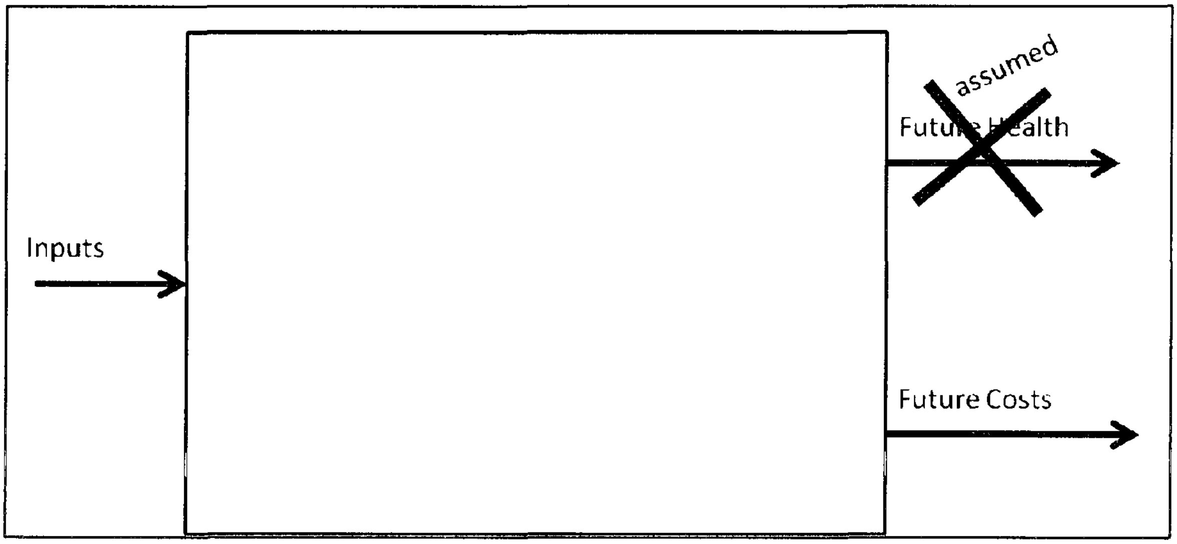


Figure 8-Perfect System Health is Assumed to be a Requirement

Before diving further into the details of this model, a discussion and an assumption will be made. The ideal level of health for systems can vary. While many systems need to be maintained in 100% working order, there are also situations where partially broken systems are acceptable. Systems can be fielded with redundancy built in where maintaining 80% of the system in working order at any given time is "good enough" and paying the additional cost to fix the other 20% simply is not worth it. For simplicity, this dissertation will focus on systems where the goal is to maintain them at 100% health. There is an inherent tradeoff between cost and health, and it will be assumed that tradeoff is set to prioritize health first. As a result, this dissertation presents a method which minimizes the cost of maintaining a system at 100% health. As such, the "Future Health" output is unnecessary since it can be assumed to be 100%. The one output now becomes focused around answering the question "How much will it cost to ensure the system continues to function at 100% as needed?"

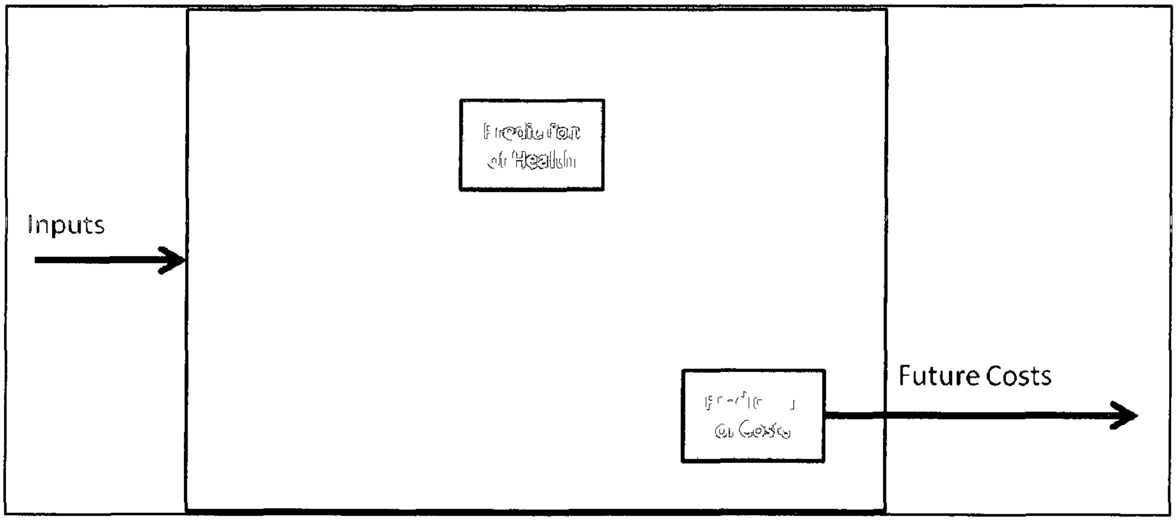


Figure 9- Costs will need to be predicted. They will be related to the health of the system.

Thus, a module will be required which can predict these costs. Additionally, since there is a requirement to maintain 100% health, the model will need to predict when challenges to that health occur. This leads to the two modules shown in Figure 9: Prediction of Health and Prediction of Costs.

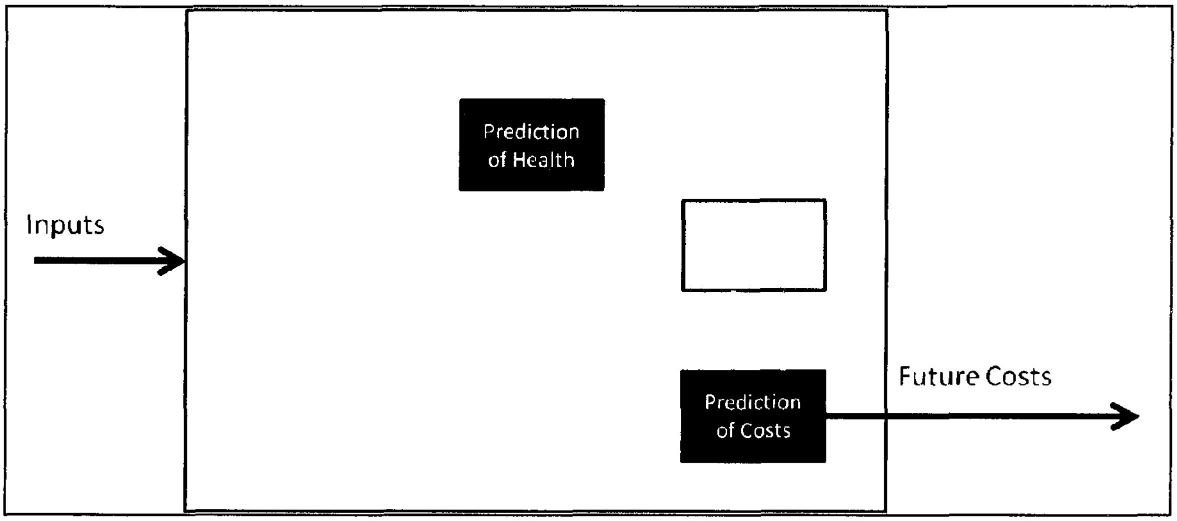


Figure 10- Cost is a Function of the Chosen Sustainment Effort, Which Depends on the Effort's Cost

The goal of the model is to create a feedback signal for decision makers based around minimizing costs. This means the final costs themselves are not necessary. Instead, the differences in costs matter. If all possible decisions result in the same $10M lifetime cost, then that $10M is irrelevant since there is no reason to pick one choice over another. Because only the deltas matter, costs which do not change are irrelevant, and only costs that do change need to be considered. In the sustainment phase, this means standard fixed costs do not matter, and only costs based on required sustainment efforts have any impact. If one scenario involves a system being perfectly healthy with no effort but another scenario involves the purchase of additional spare parts, that difference is what needs to be considered. As a result, the Prediction of Costs module will require, as an input, knowledge of what the future sustainment efforts are. Additionally, the choice of which sustainment effort will be performed depends on the expected costs. After all, if two equally valid solutions are available, the cheapest one will win out. To accomplish this, a Sustainment Effort module needs to be created, as shown in Figure 10.

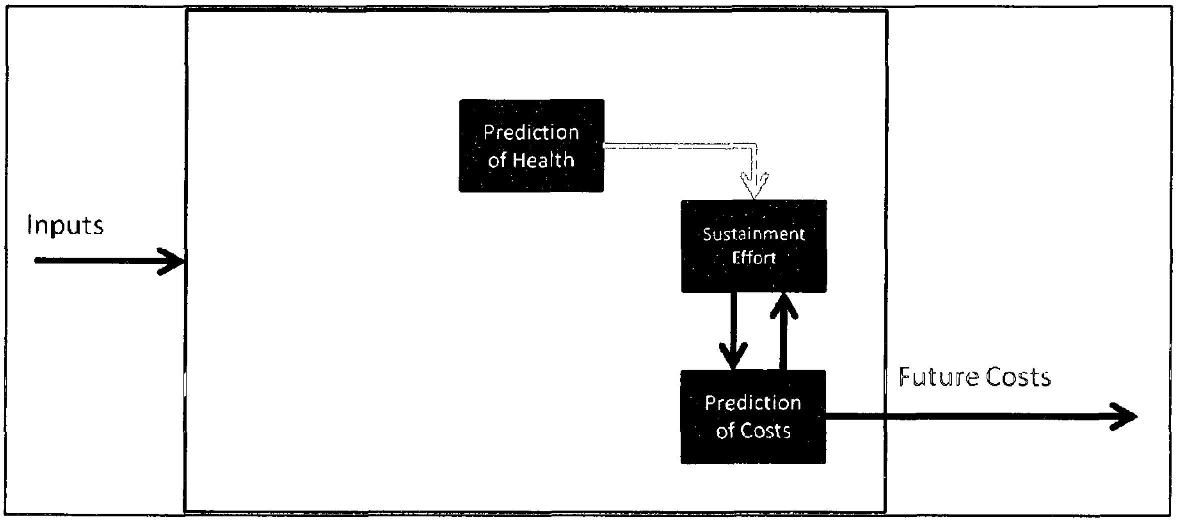


Figure 11- The need for Sustainment Efforts is Based on a Prediction of Health

The Sustainment Effort module will determine the correct effort, but it needs to be triggered. The Prediction of Health module will determine when the health of the system is going to drop below 100%, triggering the requirement for a sustainment effort, as shown in Figure 11.

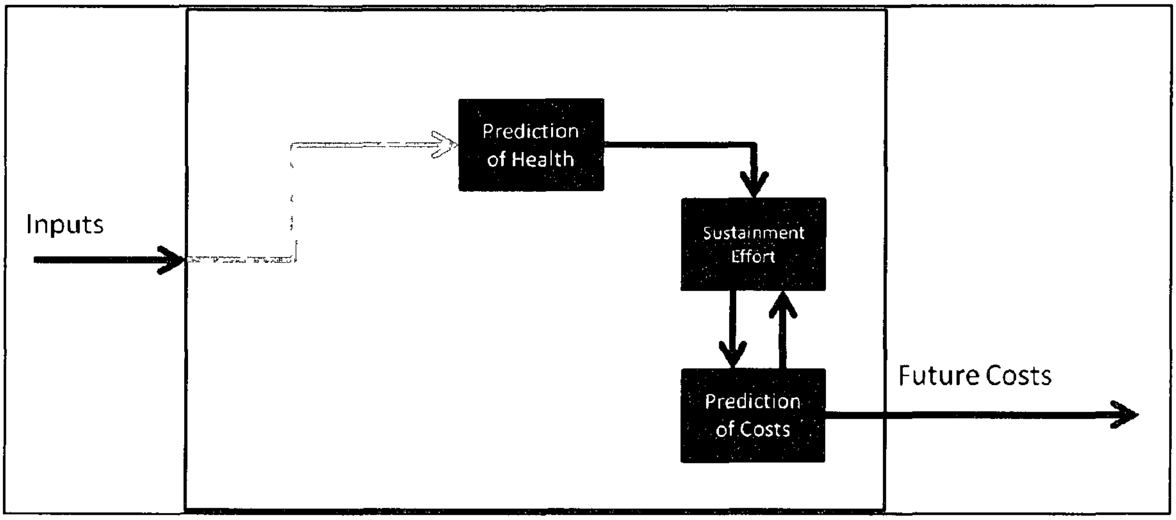


Figure 12- Completed Concept without Randomness

The model could be complete at this point, since it is possible to go from basic inputs directly to a Prediction of Health. An extremely simplified demonstration of this would be inputs showing a case where there is a surplus inventory of 10, 1 part is failing every month, and there is no ability to repair or purchase. In that case, it is simple to see that the Prediction of Health would show a problem developing in 10 months when inventory runs out. This simplified model is shown in Figure 12.

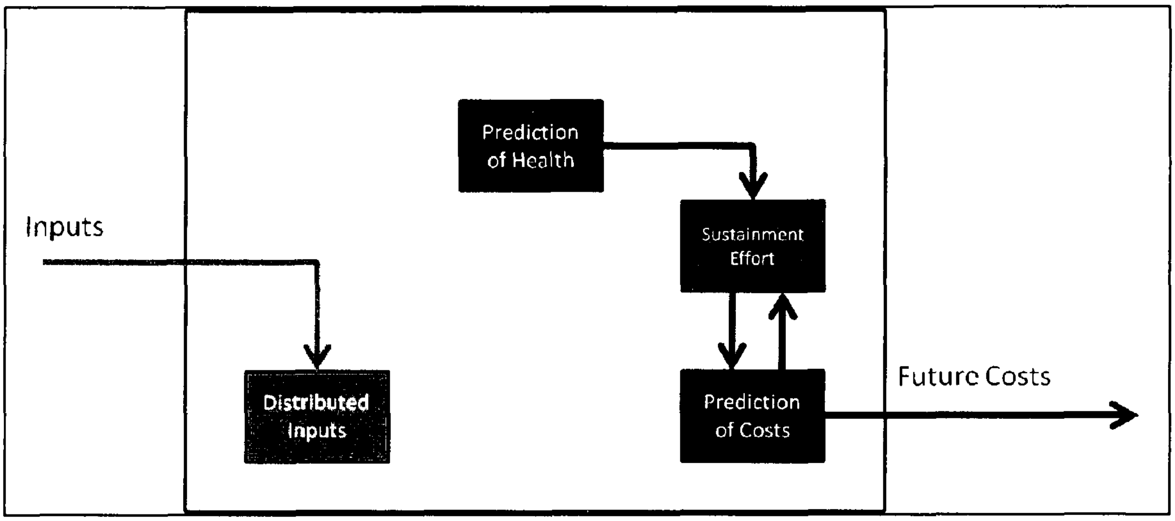


Figure 13- Adding the Distributed Inputs module to consider randomness

However, this discounts what happens in reality. It assumes our inputs are perfect, when the truth is they rarely are. A history showing an average of one part failing per month does not neccesarily mean exactly one part will fail each and every month. A vendor predicting they will repair a part for five more years does not guarantee they will do so. There is inherrent randomness in these inputs. A good model should take this randomness into consideration and turn it into something statistically useful. It is dangerous to rely too heavily on a model which can only predict the "most likely" scenario. If a system has a 20% chance of having problems in 2 years, 60% chance of problems in 3 years, and 30% in 4, we should want all of that information. If we do not take that randomness into consideration, the model for this same scenario would simply result in a prediction that the problems will occur in 3 years. This could cause us not to prepare for the possibility of problems earlier or later. As demonstrated in the introduction, it is important to consider these possibilities since the ideal solution depends on these possibilities. These possibilities determine the likelihood that the initial solutions will be insufficient and there will be consequences in the future. As shown previously, understanding the chances of those consequences can provide solutions that are significantly more efficient.

In order to account for this randomness, the inputs will need to be converted into distributions. As an example, a vendor's claim that they will produce a part for 3 years needs to be converted into a distribution that shows how likely they are to stop production at other times as well. Depending on the available information for each input, this can be accomplished either through data mining and distribution fitting or purely through appropriate mathematical reasoning.

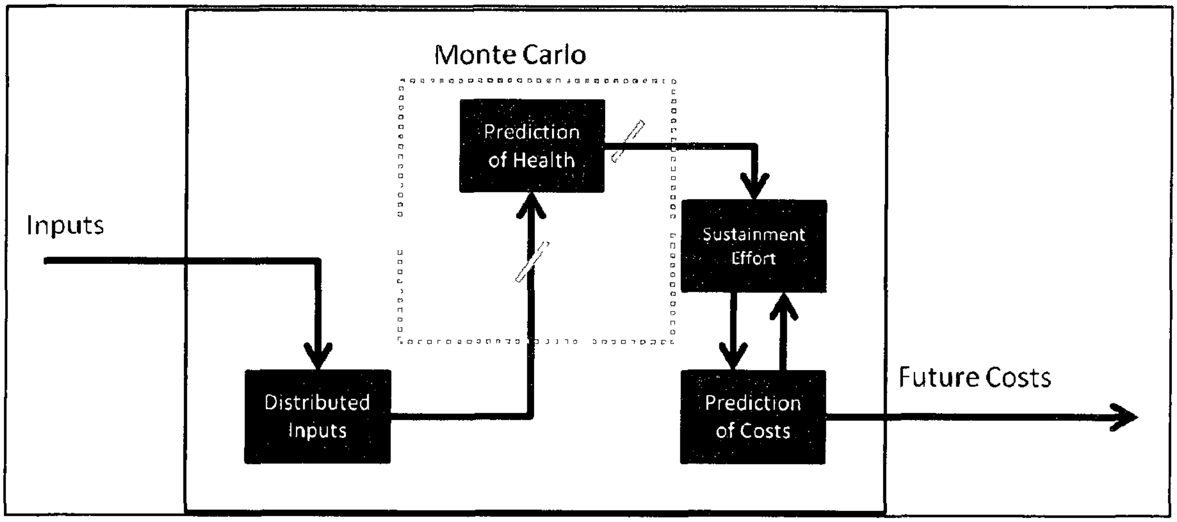


Figure 14- Completed Model

The last question is how to go from these new Distributed Inputs to a Prediction of Health. It was demonstrated earlier how simple a Prediction of Health can potentially be with single, certain inputs. However, instead of single inputs, the model now has distributions. To bridge this gap, a Monte Carlo technique is used. Samples will be taken from each distributed input and fed into the simple Prediction of Health module. This will be done repeatedly with a large number of samples from each distribution. The resulting predictions will be combined and histogrammed. For example, if 1000 sample sets are taken and combined, and 370 show no problems anytime in the future, it can be extrapolated that there is a 37% chance that no problems will occur and a 63% chance that a solution is needed.

This completes the basic concept behind the method and each of its modules. What remains is to show the current state of methods and research, the details and math behind this concept, and its effectiveness.

2.2.4 Originality of Method Existing Methods

Attempting to efficiently deal with the impact of obsolete parts is referred to as DMSMS by those who work in this area. In fact, as of the writing of this paper, Wikipedia's article on Obsolescence has a section called Obsolescence Management which contains only 27 words and a link saying "See DMSMS." The article on DMSMS, on the other hand, contains 745 words. This is just one such indication that the research in this area is dominated by the DMSMS concept.

DMSMS stands for Diminishing Manufacturing Sources and Material Shortages. As the name implies, it is concerned with two main aspects. First, the "Diminishing Manufacturing Sources" portion refers to when a part goes obsolete because no one manufactures it anymore. Second, the "Material Shortages" portion refers to when a part goes obsolete because there is a shortage of material needed to continue manufacturing it.

The very name DMSMS suggests this entire topic of research is focused around an area that does not address the modern scenarios I discussed in the Background section. The Diminishing Manufacturing Sources portion of the name comes from a time when if you wanted to build an electronics-based system, you had to design and build the entire system down to the component level. You might design a particular board and during that design you might discover that you required, for example, a specific integrated circuit in X form-factor, with Y pins, that operated at Z voltage, etc. When you initially designed the board, there might be 4 companies manufacturing such an Integrated Circuit. Over time that would drop to 3 manufacturers, then 2, then 1. Hence, the phrase "Diminishing Manufacturing Sources."

This simply is not how many systems are designed anymore. As discussed earlier in the example of the Perry Memo (Perry, 1994), the way of the world is no longer to build boards yourself but to go out and buy them from Motorola, Intel, etc.

DMSMS efforts have resulted in massive systems dedicated to making sure that, if we need to, we can produce a board for 20 years. But the only ones who want a board produced for 20 years are those who are fielding a system for 20 years, and these days those people are not the same ones building the boards. The ones who are building the boards, the Motorolas and Intels, have found the most effective business practice is to produce these boards for only a few years before moving on.

In summary, the builders of these systems are not running into problems because DMSMS is preventing board designers from producing boards for 20 years. They are running into problems because board designers are not interested in producing them that long even if they could. Understanding this leads to a fundamental shift in how to support a system.

When you control your own design and production of boards, you fight obsolescence so that the boards will be produced as long as you need them. You build systems to keep DMSMS in check in order to keep producing.

When Motorola and Intel build your boards, however, you have to accept the fact that they are only going to be produced for a few years. When fielding a long-term system, you do not fight obsolescence with DMSMS techniques because that is inherently a losing battle. Instead, you accept that the boards in your system will go obsolete and you set about finding the most efficient way to keep sustaining an obsolete system.

This shift just barely appears to be starting. There is a tremendous body of work available which discusses preventing DMSMS impacts. When researching this area, most of what I was able to find was DMSMS related. However, there is comparatively very little work dealing with the new reality where you accept obsolescence of boards and find ways to continue on despite it.

DMSMS solutions are becoming increasingly irrelevant. What we have tended to do with them is use them to ensure the continued availability of parts, and to use this continued availability to adopt a "just-in-time" sparing method which minimizes inventory (Howard, 2002) and therefore, supposedly, cost as well. What we need instead is a system dedicated to the new reality, which is that our boards will only be produced for a short time regardless of our desire. We either have to perform solutions which let us keep operating a system years beyond when a board goes obsolete, or we must plan for our systems to constantly have their hardware upgraded to newer products.

The FROST Method accomplishes this. It lets you determine, on a part-by-part basis, when the most efficient solution is to stockpile parts while they are available, when the most efficient solution is to plan an upgrade, etc. With this background in mind for my research, I set about trying to discover products and research which were taking a similar perspective to solve this issue.

A scouring of available products at the moment shows that most of them are based around the old way of doing business. The few that have adapted to help make decisions based around obsolescence do not consider many of the issues one would need to consider to make an informed decision about the right solution. The products are shown in Table 3.

Search Term	Results	Link
"Total Parts Plus"	13,700	www.totalpartsplus.com
"obsolescence management information system"	13,400	
QinetiQ Q-Star	5,340	www.qtec.us/
ARINC Obsolescence Management	4,520	www.arinc.com/products/dmsms/index.html
NSWC Crane Horizon	4,290	
"Future Readiness and Optimized Scheduling Tool"	198	
"Supportability Management Assessment Report Tool"	180	www.mysmart-rac.com/rac/smartProduct.htm

Table 3 - Selected DMSMS Tools and Their Relevance Based on Google Search Results

These tools vary considerably in functionality. Total Parts Plus, for example, at its heart is a database tracking when components will go obsolete and whether there are other vendors manufacturing identical parts. OMIS, on the other hand, incorporates inventory, failure rates, usage, etc. to make a prediction of the future.

One shared trait amongst these tools, however, is that none of them truly consider uncertainty. If a part is going obsolete, it will go obsolete on exactly that date, with no variation. If inventory is expected to run out, it will run out on a specific date, with no variation.

This philosophy inherently means none of these tools is currently able to attempt the proposed method. At the core of the proposed method is the concept that you can reduce your chances of running into problems later by spending more now, or you can increase your chances of running into problems later by spending less now, and that finding the right balance could save a lot of money. Since these tools deal in concrete certainties, they can only consider that the chance of running out of inventory is binary; either there is enough inventory, or there is not. This makes them fundamentally unable to optimize.

Since there is apparently no real-world option available to accomplish this method, the next question is whether this has been previously discovered in research. A review of literature shows four significant areas of research related to improving the efficiency with which we sustain engineering systems. In addition to these four groups, there is one additional relevant piece of research which will be discussed later.

The first group is concerned with improving our knowledge of when parts will go obsolete. This is certainly an important topic and there is a large amount of research on the subject. However, these methods do not directly address the same issues as the FROST method. Having knowledge of when a part will go out of production or support is key, and in fact this knowledge is one of the inputs required for the FROST Method, but simply knowing when a part will no longer be available for purchase does not tell you the best way to support a system. Generally these methods are provided as a starting point for systems to use as a way to determine where they should be focusing their attention. A few go further and use their predicted obsolescence as the basis for a sustainment plan. For example, Herald et al. propose a method to use technological obsolescence curves to predict the frequency with which parts will go out of production and new parts will become available (Herald, Verma, & Lechler, 2007). They then use basic cost information and inventory to suggest when parts should be substituted with newer versions, as well as when parts should be stockpiled in order to delay having to perform a substitution. This method is a long way beyond what is currently being used in industry by most groups. However, it does include the inherent assumption that systems will have a philosophy of continuously performing tech refreshes of all obsolete parts in order to maintain supportability. Many systems do not share this philosophy because they value the stability of having known parts in their system and are more than willing to explore other solutions to extend the time with which they can support obsolete parts instead of moving onto newer parts which might introduce bugs and unknowns into their systems. As a result, Herald's method is situational. However, in those situations it would likely be quite effective. A worthwhile area of continued research would be to combine that method with the FROST Method. While the FROST Method considers a much wider variety of factors and other solutions, its handling of substitutions could likely be improved by incorporating some of the principles put forth by Herald, Verma, & Lechler.

The second group is concerned with finding an optimal general maintenance philosophy. For example, in 2007 Wang, Chu, and Wu discussed using fuzzy logic to select between maintenance strategies, such as time-based maintenance, corrective maintenance, or predictive maintenance (Wang, 2007). Similarly, in 1997, Frangopol, Lin, and Estes proposed a method to determine an optimum inspection/repair method to help minimize life-cycle costs of a system (Frangopol, 1997). While these papers talk about many of the same inefficiencies discussed in this paper, their approach to improving efficiency is focused on discovering when maintenance should be triggered in order to maximize availability and production. These tend to be focused on industrial systems featuring mechanical parts which wear down over time. Mechanical parts feature a bathtub curve (Wilkins, 2002) where a part's failure rate increases near the end of its lifetime due to wear-out. Maintenance can alter this bathtub curve, extending the lifetime of a system and reducing the rate of failures. As an example of maintenance of a mechanical part, consider when you change the oil in your car's engine to keep it running smoothly. These papers present methods to optimally balance various factors through maintenance philosophies, but they fundamentally do not apply to electronics-based systems because there is no equivalent to changing the engine oil on a circuit board. Electronics generally have a constant failure rate which is independent of time. Maintenance philosophies serve to alter or delay the wear-out phase of a part's life-cycle. Since wear-out phases are not a factor for electronics, this group of methods is inapplicable. As Dennis Wilkins puts it, "For many electronic components, wear-out is not a practical failure mode. The time that the product is in use is significantly shorter than the time it takes to reach wear-out modes" (Wilkins, 2002).

The third group, which seems to be a much smaller group than the other three in terms of the amount of available research, is concerned with making hierarchy-based decisions (Bevilacqua & Braglia, 2000)(Beck, 2003). For example, one paper states that

> "The preferred order of potential solutions, beginning with the lowest-cost approach, is as follows:
>
> 1. Life-of-Type buy

2. Reduce part performance requirements
3. Find another source
4. Develop another source
5. Emulation
6. Redesign
7. Cannibalization" (Beck, 2003).

These papers simplify by removing one of the fundamental concepts of this dissertation; the choice of the best solution differs on a part-by-part basis. These papers generally acknowledge that this hierarchy is not always correct for every situation, but for simplicity's sake they operate as if it were. While not as common in research, this method is, in my experience, the most common in actual use. The assumption that a Life-of-Type buy is the best and therefore default solution is fairly common. As the FROST Method's results will show later on, there is significant efficiency to be gained by not making this assumption.

The fourth group, and the closest to this work, concerns finding the optimal time or situation in which to upgrade or replace a system. The problems stated in these papers are extremely similar to those claimed in this dissertation. For example, Mummolo states that "A lack of usable models leads to procurement which is premature, inappropriate or simply unnecessary" and sets out to find "the cost expression of all the events which could influence the life cycle cost of a device" (Mummolo, 2008). Sandborn and Singh go so far as to offer a tool called MOCA that was not mentioned in the previous section, which is focused on helping program managers determine when obsolescence will cause a system or part to become unsupportable (Singh, Sandborn, Lorenson, & Geiser, 2002). These papers, however, focus only on determining when to upgrade and replace. There is minimal help provided to determine how to actually get to that time when a system or part will be upgraded or replaced. They might do an excellent job of telling managers that they should replace their system in 28 months, but they will not tell them whether they should support the system's processors during those 28 months by purchasing more parts, setting up repair contracts, qualifying a replacement, etc. Since the choice of solutions in the meantime is not even considered, there is no discussion about how to optimize these solutions.

These four groups account for the vast majority of related research available. There is, however, one paper and associated tool which fits much closer in line with the FROST Method proposed. It attempts to use a similar concept to optimally solve a single solution type: purchases. In 2007, three researches at the University of Maryland presented a paper that "extends the final order model for machine equipment and applies it to the electronic part obsolescence problem. The only other known quantitative treatment of lifetime buy optimization for electronic parts is by Rugina, which discusses various models for lifetime buy quantity determination without implementation" (Feng, 2007) (Rugina, 2000). Their proposed method shares several things in common with the FROST Method. It utilizes distributions and a Monte Carlo technique to attempt to optimally determine a solution for maintaining a system beyond obsolescence dates. However, while it discusses the possibility of being used in the same sorts of scenarios being addressed in this paper, it is not really a complete solution in those scenarios. Their choice of case study reveals the ideal scenario for their proposed method: production of electronics.

In producing an electronic device, if a component such as an integrated circuit chip goes obsolete, or if a last batch of the device is being made, the vast majority of the time the best solution is to stockpile the components expected to be needed in the future. As a result, in this scenario it is a relatively good assumption that the best solution type is to perform a purchase and then use their method to determine exactly how large that purchase should be. This is related to the hierarchy methods mentioned earlier. The assumption is that a purchase is always best.

In sustaining a system, however, it is often not safe to make the assumption that the best solution type is a purchase. The best solution often involves repair and reuse, among other possibilities. If somehow it can be determined that the sole solution should be to make a purchase, then their proposed method becomes useful in this scenario. However, this still leaves the requirement to A) identify the best solution in the first place and B) optimize that solution if it is anything other than a purchase. Additionally, they acknowledge that forecasting demand is a critical issue filled with uncertainty, but they scope it out of their method with the statement that “it is assumed that demand forecasts are supplied from another source.”

Overall, my literature review showed this to be a fairly unexplored topic. The only notable closely-related work I was able to find was published only 3 years ago and that paper takes the time to mention that the only related research they could find does not even discuss implementation (Feng, 2007). That method of using Monte Carlo to optimize a solution is a good one and shares much in common with the FROST Method, but it is only one small piece of the puzzle. A method that only considers a single solution is limited in what it can be applied to. A method which considers multiple solution types and has broader applicability does not seem to be available, either commercially or in research. As a result, the method proposed in this dissertation appears to be novel.

2.2.5 Validation of Methodology

The model behind the FROST Method is composed of several logical modules and linkages. Ideally, if it could be shown that every module and every linkage is error-free and has 100% confidence in being perfectly accurate, the model would also be shown to be error-free and 100% accurate. While it is not practical to perfectly find and remove all inaccuracy from a model which heavily relies on randomness, steps can be taken to minimize the inaccuracy. A combination of logic, assumptions, and problem scoping will be used to demonstrate that many of the linkages and modules do not have any significant way to contribute to inaccuracy. This validation effort will result in three remaining potentially significant sources of imperfect correlation which cannot be easily removed. Two of these sources will be combined and data will be run through them via simulations to demonstrate the effectiveness of their combined output. A high confidence will also be demonstrated for the last remaining source of imperfect correlation via simulations. Having reduced the model to two outputs without any dependencies which would cause the confidence to drop when combined, these two outputs will then be mathematically combined and the result will show that, even with the sources of imperfect correlation accounted for, the FROST Method significantly outperforms current methods. This performance, combined with the causality demonstrated in the construction of this model, will serve to validate the method.

The first step of this validation process is to start with the model developed earlier.

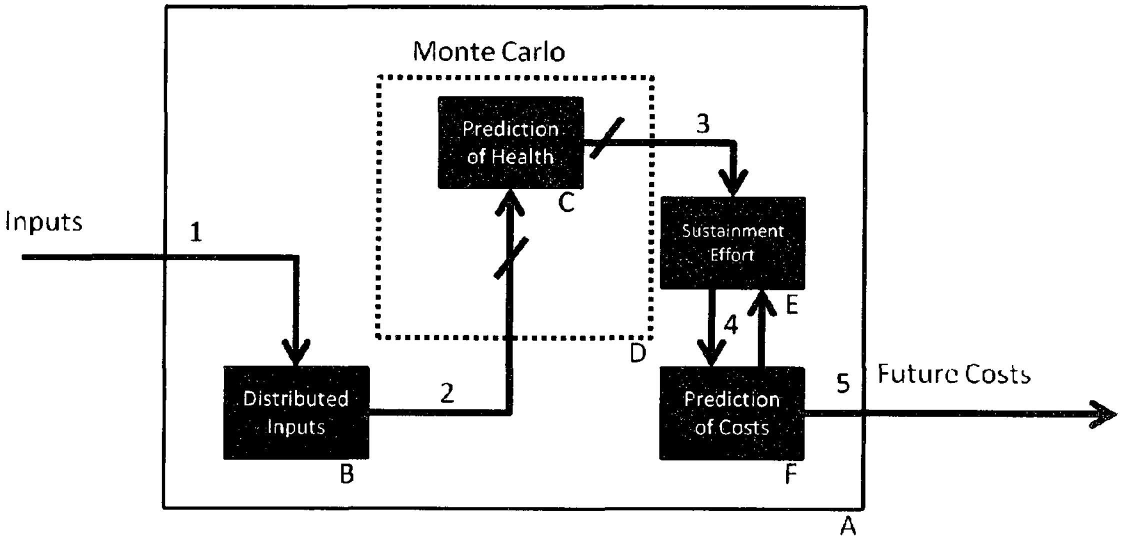


Figure 15- High-Level Method with Labels

Figure 15 shows the method with each linkage and module labeled. The correlation for the overall model, C_A, is a function of the individual correlations inside C_A's boundary.

$$C_A = f(C_B, C_C, C_D, C_E, C_F, C_1, C_2, C_3, C_4, C_5) \tag{2.2}$$

If all linkages and modules were independent, the correlation for the overall model, C_A, would be as follows.

$$C_A = C_B \times C_C \times C_D \times C_E \times C_F \times C_1 \times C_2 \times C_3 \times C_4 \times C_5 \tag{2.3}$$

Each one of these variables will be considered and analyzed, and many of them will be logically demonstrated to have nearly perfect correlation. Variables with perfect correlation can be removed from these equations as they do not have any impact.

First, some clarification of C_A is required. While the model inside this boundary could be used for a wide variety of purposes, the claim of this research is that it will be used to compare two fielding schedules and provide an estimate of the difference in cost between those two scenarios in order to determine which is the better choice. As part of this cost estimation, the model will provide a best-solution for each part which it expects engineers and logisticians to follow. This will provide a further improvement over current methods since the proposed solutions are expected to be as or more efficient than the ones found using current methods.

The variables representing the confidence in the model's linkages will be considered first. These linkages represent data transferring between modules. For these links to be valid and to avoid introducing inaccuracy, two concerns need to be satisfied. First, they need to provide sufficient data for each module to function correctly. Even if every module were ideal, the overall model would not function if those modules were not receiving the inputs they need to function. Second, the process of transferring data from the output of one module to the input of another cannot somehow introduce any error. If these two concerns are satisfied for a linkage, then it is clear that that linkage does not in any way reduce the model's correlation with reality.

Linkage 1 represents data being inputted into the model. This linkage can introduce a lack of correlation to the model if the data entered is either inaccurate or insufficient to run the model. Errors due to insufficiency can be removed by creating a basic requirement for using the model that all required inputs will be provided to the model. In a practical implementation of the FROST Method through software, this can be guaranteed by having the software refuse to perform the method if a key input is missing. For purposes of this research, however, this will be dealt with by making an assumption that the overall model will be provided with sufficient data. A number of such assumptions will be made in this section; for reference, they are summarized at the end of this section as well as again in the appendix.

The remaining question with Linkage 1, then, is whether the data inputted into the model is accurate. Knowingly, it is not. This will be dealt with in two ways, depending on the characteristics of the input. The first way is reserved specifically for inputs relating to fielding schedules. Future schedule data is likely to be inaccurate because schedules tend to be fluid, with systems being fielded or retired earlier or later than originally planned. This is a special case of inaccuracy. Unlike other inaccurate inputs, it is not the case that this is a variable completely beyond control which simply cannot be guessed at accurately. Instead, schedule is generally within management's control, but management makes conscious decisions to deviate from initial plans for various reasons. For example, management might choose to delay fielding a system in order to gain extra time for engineers to improve the product.

With an input such as end of production dates, it might be necessary to run simulations with 10 different possible dates because there is simply no way to have any idea which nine dates are wrong and which one date is correct. In comparison, if a decision is made to simulate 10 different fielding schedules to determine the best one, it is not the case of nine inaccurate schedules and one accurate one. It is, instead, the case of 10 potentially accurate schedules, nine of which will likely later be dismissed by conscious choice. This is an important paradigm to understand because it means schedules should not be treated as a random variable. The feedback loop included in Figure 6, the high-level methodology, causes schedule to not be a random variable. The model provides feedback on the different schedules and this impacts which schedule will become true. Compare this to other variables such as end of production dates. Under normal circumstances, the output of model will not have any effect on when a part stops being produced, so as far as the model is concerned this is a truly random variable. Thus, an assumption can be made that the fielding schedules entered into the model are either accurate or are potentially accurate, and therefore there is no need to worry about their impact on the model's correlation.

Other input data, however, is inaccurate in a way the users of the model will have no control over. Historical measures of failure rates do not always correspond perfectly with future failure rates. Vendors do not always produce or repair parts as long as they claim or expect. These predictions of future data, although based on the best current data available, are often incorrect. These situations are a genuine concern, and they will in fact lower the correlation of the model with reality to well below 100%. However, this model already contains a module whose sole job is to attempt to deal with these inaccuracies, the Distributed Inputs module. The lack of correlation that will be caused by this issue does need to be considered, but it only needs to be considered once. In this case, I can choose to consider it as part of this linkage and have it covered by the correlation variable C_1, or I can choose to consider it as part of the Distributed Inputs module and have it covered by the correlation variable C_B. I choose to cover it in C_B. As a result, C_1 will be considered to have essentially perfect correlation, although it has already determined that C_B will not.

Next, C_2 is the correlation represented in the link between Distributed Inputs and Prediction of Health. This link's function is to randomly sample the distributions in the Distributed Inputs module and provide those samples to the Prediction of Health Module. The danger with sampling is that if it is not done correctly, the randomness inherent in Monte Carlo will allow the same analysis to be performed twice with different results. Two different predictions of the future cannot both be true, so this proves that at least one of these futures must be incorrect and that inaccuracy has made its way into the method. A good model should have results which are stable and repeatable. This depends on the sample size. For example, if an attempt is made to try to determine the probability of a coin flip resulting in heads by performing only two samples (flips), one quarter of the time the result will be 100% heads, and one quarter 0% heads, both of which are quite wrong. On the other hand, if a million coin flips are performed, there is a better than 99.99% chance of getting a result that shows heads has a likelihood between 49.98% and 50.02%, which is very accurate. So, the correlation of C_2 depends on whether the sample size is sufficiently high. Perfect correlation based on randomness is not possible to guarantee; no matter how large the number of coin flips is made to be, it will never get sufficiently large for there to be a guarantee that exactly 50% heads is achieved. However, a sufficiently large sample size can virtually guarantee that the impact of this imperfect correlation is negligible.

An equation for the sample size required is difficult to provide since it depends on the specific parts and distributions being used for a system. However, once an implementation of this method is created, it is simple to determine whether a sufficient sample size is being used. With all other variables being held steady, if multiple simulations are performed that give different results, then that difference is due to the randomness of the Monte Carlo process and can be adjusted by changing the sample size. Implementers should determine a threshold for their system. For example, one could determine that differences of $100 are small enough that they will not affect anyone's actions. If repeated simulations are run and the result varies by less than $100, the sample size is large enough since it has no effect. If the results vary by more than $100, the sample size needs to be increased. In this manner, the method can be tuned to eliminate the randomness' practical impact, essentially raising its correlation to 100%. A requirement for the practical implementation of this method will be to tune the sample size. With this requirement in place, for validation purposes it can be assumed that the sample size will be sufficiently high to create repeatable results. This causes C_2 to be essentially 100%, since the error approaches zero as the sample size increases and the sample size can be set as high as is needed.

C_3 is the correlation representing the link between the Prediction of Health module and the Sustainment Effort Module. As previously mentioned, this linkage serves to trigger a sustainment effort, so for it to be valid and correlated it needs to be shown that the outputs from the Prediction of Health are sufficient to identify any problems that are in need of solutions, and also to provide the Sustainment Effort module with enough detail about those problems so that it can determine which solutions are valid and which are not. There are two possible kinds of sustainment efforts, and these scenarios will be validated separately. The first is for proactive sustainment efforts, which are enacted before a problem has actually occurred. The second is for reactive sustainment efforts, which are enacted after a problem has occurred. It will be shown that for both of these types of effort, the output of the Prediction of Health module is a sufficient trigger.

For proactive sustainment efforts, managers must rely on something triggering them that there is a need for action before an effort will be started. When that triggering occurs, they begin a sustainment effort to avoid the problem. If the triggering in the model and the triggering in the real world match up, this has perfect correlation. The expectation is that the processes in the FROST Method will actually be used by those managing the systems. In other words, when a problem triggers in the model, that trigger will result in an output which recommends a solution, which will in turn trigger system managers that action is required. Since that same trigger is shared, there is perfect correlation as long as engineers and logisticians are actually using the FROST Method to solve problems.

A reactive sustainment effort would occur in extreme cases where a problem is not predicted in enough time to respond before it hits. According to earlier assumptions and definitions, this occurs when inventory is zero and a part fails. This can be anticipated to be a very rare occurrence with this model in place, but it is possible. As an example, the model could show that inventory of one will be sufficient in 99.9% of future scenarios, and as a result only one extra part is kept on hand. If a part fails the model would show for someone to again raise inventory levels to one spare, triggering a proactive effort. But what if, in the rarest of situations, another part failed before additional inventory had been procured? That would put the system sustainment into a reactive situation because inventory would be zero with a failed part. In this scenario, the model would still show a future negative inventory. In fact, it would show an immediate negative inventory. Since a sustainment effort is triggered whenever future inventory is shown to be negative, this reactive scenario would successfully serve to trigger a sustainment effort. Thus, even in this situation the model would still successfully trigger.

Overall, this gives C_3 ideal correlation as long as the FROST Method is being used to make decisions. In proactive scenarios, the model would match up with the real-world trigger because it actually *is* the real-world trigger. In reactive scenarios, the model would still pick up the problem in the same way, triggering any user to start looking for a solution immediately.

C_4, as shown in the model, is a two-way link connecting the Sustainment Effort Module with the Prediction of Costs Module. The purpose of these links is to allow these two modules to interact in such a way that they can determine an actual sustainment cost. The Sustainment Effort module determines viable solutions. The Prediction of Costs module predicts the cost of each of those solutions. Knowing the cost for each viable solution, the Sustainment Effort module can pick the best one and feed that information back to the Prediction of Costs module, which outputs a final cost. With that concept of operations in mind, each direction of this link will be discussed individually.

First, one directional link shows the data traveling from Prediction of Costs to Sustainment Effort. This link provides the Sustainment Effort module with an accurate cost for each viable solution, which it then uses to select the most efficient (cheapest) option. In order for this link to be successful, it needs to be providing A) accurate cost predictions for B) all viable solutions. Part A is the very function of the Cost Prediction module, so the correlation for that will be addressed during the discussion for the correlation of the Cost Prediction Module, C_F. That just leaves part B. This means that the link from Prediction of Costs to Sustainment Effort will be accurate if the Sustainment Effort is provided an accurate list of all viable solutions. Since providing this list is the job of the link going in the opposite direction, I can state that this direction of the link is valid and has direct correlation if and only if the link in the opposite direction successfully provides all viable solutions.

Next is an analysis of the link in the opposite direction, from Sustainment Effort to Prediction of Costs. It has already been determined that for this bi-directional link to have perfect correlation, this link needs to provide all valid sustainment solutions. Additionally, it needs to provide a sufficient level of detail for the Prediction of Costs module to function.

In order for this link to provide all valid sustainment solutions, the model will need to initially be provided with knowledge of possible solutions as well as a feedback mechanism. With the module containing all possible solutions, it is then the job of the Sustainment Effort module to determine which ones are valid or not, which will be discussed in that module's section. With this setup, the only worry is if a new solution type becomes available. This is an extremely rare event, but it does occur every so many years. To cover this, a feedback mechanism will be included which will be able to adapt and add in any new solution types which exist in the future. This means imperfect correlation will only be caused in this area during times when a new solution type is available but has not yet been incorporated into the model. This will be discussed in more depth later on, and its error will be covered as part of the correlation for the Sustainment Effort module, C_E.

The last requirement mentioned is that the link should provide a sufficient level of detail about each solution for the Production of Costs module to function. This will be validated by scoping and assumptions. Realistically, there is a seemingly infinite number of variables that could affect the cost of a solution, including economic measures, commodity prices, unemployment, etc. Instead of worrying about these variables, this research will scope the "Future Costs" outputted by the overall model to only contain costs that are directly affected by inventory level, part type, date, and a few other basic variables. One key reason for that is that the real goal of this model is to determine differences in cost due to schedule changes and solution choices. While many variables affect cost, most of them can be expected to be only negligibly affected by schedule and solution; therefore, they can be ignored. Furthermore, there are no claims being made that this model can accurately be used for detailed accounting. In order to operate it needs to have an understanding of the delta in costs between two scenarios, but not necessarily the actual costs themselves. In other words, it is not important to know that option B will cost $10M, it is important to know that option B will cost $1M less than option A. For simplicity, an assumption will be made that variables other than the ones scoped into this model will affect all scenarios equally and therefore not have an effect on the delta between scenarios. So, under the assumption that all relevant future costs are covered by this method, all requirements for this bi-directional link have been discussed and all significant sources of imperfect correlation have already been assigned to other correlation variables (specifically C_E and C_F), giving C_4 practically ideal correlation.

With the correlation for the links considered, it is time to move to the correlation for the modules. C_B is the correlation for the Distributed Inputs module. This module simply converts an input into a statistical distribution representing that input. This will occur in one of two ways. The first way is by using known mathematical relationships. For example, there is a demonstrated relationship that can take the number of hours of use for a part, along with the number of failures witnessed during that time, and convert it into a distribution of likely true mean-time-between-failures (MTBFs). For this input, the fact that a proven mathematical relationship exists will serve to validate this module. For other inputs, however, distributions will be created by collecting a statistically significant number of data points using historical data and performing a distribution fit. These distributions will be imperfect, but since they are based on real data, it will be possible to measure this imperfection and provide actual values for the sources of error that make up C_B. Specifically, both Kolmogorov-Smirnov test scores and Chi-Squared statistics will be provided to describe the correlation the found distributions have with reality. Much of the error in the overall proposed method will come from imperfect inputs which are addressed in this module, so this module can be expected to have the lowest correlation. This imperfect correlation will be included as part of the simulations, so if the simulations still result in improved efficiency over current methods then it will be apparent that this module's correlation is not a significant enough issue to harm the overall method. If anything, it will be an encouragement as it means that there is further room for improvement and anyone who wanted to put the effort into improving the correlation of this module would be rewarded with even better performance than will be claimed in the results section of this dissertation.

Next, C_C is the correlation for the Prediction of Health module. This module relies on a key assumption made previously: that problems with system health can be determined by identifying when a part will fail and no inventory will be available to immediately replace it. This makes the module relatively simple. Anytime a system is operational, there is a risk of a part failing. This means that to determine when inventory is zero and there is a risk of a part failing, all this module has to do is predict inventory levels and determine when they drop to zero. This can be accomplished with a very basic "conservation of inventory" approach. At any time,

$$\text{Inventory} = \text{Starting Inventory} - \text{Losses in Inventory} + \text{Gains in Inventory} \tag{2.4}$$

This gets broken down a bit further. For example, gains in inventory can come from purchases, repairs, or harvesting of parts from other systems. This conservation principle remains straightforward and does not have any room for error. If accurate values are provided as input, losses and gains can be correctly predicted and accurate values will be outputted. While it is true this module will in reality not make perfect predictions of the future, that imperfection will be due to inaccurate inputs, not due to the module itself. Most of these incorrect inputs have already been addressed in the correlation for the Distributed Inputs module, C_B. Additionally, these imperfect inputs are the same as those used in comparative methods since this is a difficulty with the nature of the problem itself and not specific to the FROST Method.

There is one inaccurate input, however, which has not already been completely addressed. Ideally this module would have perfect data about when parts will fail in the future, but since that data is unavailable it must be approximated using the best available data. In the implementation of this module, losses will be calculated based on inputs which provide A) hours of usage for each part, and B) failures witnessed for each part. This is insufficient information to perfectly predict failures. From this, an MTBF can be calculated which could accurately determine the number of failures to be seen over an infinite timeframe. For a limited time frame, however, there is randomness which will cause some imperfection. For example, if a part has an MTBF of 1000 hours it does not mean that for every period of 1000 hours there will be exactly one failure; it merely means that the *mean* number of failures will be one during any 1000 hour period. The actual number of failures could very well be zero or two. This means that C_C has imperfect correlation due to MTBFs being an imperfect predictor of future failures, which will be further discussed later.

Next, C_D represents any additional loss of correlation that might result from the Monte Carlo process. There are four basic steps to Monte Carlo:

1. Define distributions of inputs.
2. Generate inputs by sampling the distributions.
3. Calculate using the inputs.
4. Combine the results of multiple calculations.

All four of these steps are covered by other notations, which is why C_D and the Monte Carlo process are noted in the figure as a dashed boundary instead of as a full module. Step 1, define distributions of inputs, is the job of the Distributed Inputs module. Any associated correlation is covered already by C_B. Similarly, Step 2, generate inputs by sampling the distributions, is the responsibility of the link from the Distributed Inputs module to the Prediction of Health module and is covered by C_2. Step 3, calculate using the inputs, is the role of the Prediction of Health module and is covered by C_C. Lastly, step 4, combine the results, is the purpose of the link from the Prediction of Health module to the Sustainment Effort module and is therefore covered by C_3. This shows that the entire Monte Carlo process is incorporated into this model and all room for imperfect correlation is sufficiently covered by the various correlations already discussed.

Next, the correlation for the Sustainment Effort module is covered by C_E. The job of this module is to consider all possible types of sustainment efforts that could successfully solve a problem identified by the Prediction of Health module. This is based on simple logic and, with the aid of a few assumptions, should not introduce additional inaccuracy under normal circumstances. For example, using an assumption that a part is purchasable if and only if a vendor is still willing to sell it (which excludes the use of secondary markets), buying additional spares *would* be an option as long as it is done before the vendor's end-of-purchase date, and buying additional spares would *not* be an option after that date. This results in a very simple calculation to determine whether this option is actually a viable solution. These simple logic rules do not leave much room for inaccuracy in determining whether a solution is viable or not. Similar logic can be applied for essentially any solution type. The only exception is if a solution type is not considered because it is unknown to the module. Under that circumstance, unforeseen inaccuracy could be introduced because the model might not even consider what turns out to be the best solution. This will be countered by providing a feedback mechanism that allows the model to be updated when new solution types are discovered. An assumption is required that when engineers learn of a new solution type it will be added to the model. This solves the problem because when engineers do not know about a solution type, it will not be used. Therefore, the model will be correct in not considering it. When the engineers become aware of a new solution type and start considering it so will the model. Additionally, this is not a significant concern since the addition of new solution types is an exceptionally rare occurrence. For example, the last time a new solution type was introduced for Navy systems was in 2003 when Sunset Supply Base was introduced in Michael Barkenhagen's Master's thesis (Barkenhagen, 2003).

Last is the correlation for the Prediction of Costs module, C_F. As this module must be partially created on a system by system basis, its value will differ from application to application and will be discussed in more detail later in this dissertation

With all linkages and modules considered, it is time to revisit the initial correlation function.

$$C_A = f(C_B, C_C, C_D, C_E, C_F, C_1, C_2, C_3, C_4, C_5) \tag{2.5}$$

Seven of these correlations (C_D, C_E, C_1, C_2, C_3, C_4, C_5) were able to be removed by demonstrating they had ideal or practically ideal correlation. That ideal correlation relied on several assumptions:

- Problems with system health can be identified entirely by determining when a part will fail and no inventory will be available to immediately replace it.
- Sufficient data will be provided to the model for it to operate.
- Schedules provided to the model are the most accurate available. The sample size will be set sufficiently high to create repeatable results.
- The model is used by engineers and logisticians to determine when they should take action.
- Relevant future costs can be based entirely on inventory level, part type, date, and other variables which are part of the model. Future costs which cannot be determined solely by these variables are irrelevant.
- The viability of solutions can be accurately determined using simple logic.
- When a new solution type is discovered, it will be added to the model before it starts being used.

With these assumptions and requirements, the correlation function reduces to

$$C_A = f(C_B, C_C, C_F) \tag{2.6}$$

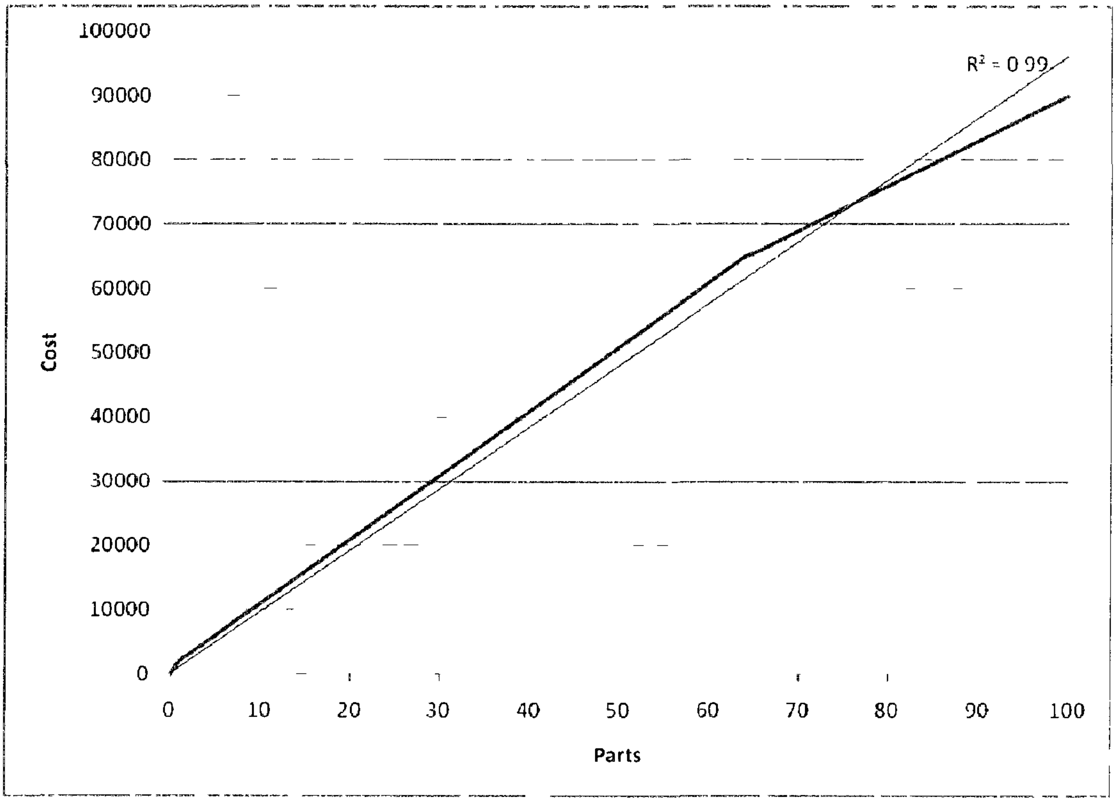
where C_B is due to inaccurate inputs, C_C is due to MTBFs being an imperfect predictor of individual failures, and C_F is due to yet-to-be-discussed errors in predicting costs.

C_F is only slightly dependent on C_B and C_C. C_B and C_C both affect the number of spare parts that are estimated to be required in the future. The number of spare parts required in the future does impact which solution will be used; therefore, there is some dependence between these variables. For example, if only one spare is needed, the solution might be to go buy one part, where if 100 parts are needed, it might be less expensive to substitute another part or perform a redesign, compared to buying 100 parts. As a result, C_F is not independent of the other errors. This suggests that these sources of imperfect correlation cannot be measured separately and then combined. However, as will be shown, in practice it turns out that C_F is quite close to independent and can therefore be calculated separately.

If cost were directly proportional to the number of spares required, C_F would be independent of C_B and C_C. To understand why this is true, assume that C_F is such that, in a particular scenario, the cost model is mistakenly making a prediction that solving a part through purchase will cost 10% more than it actually will. In other words, if one part is purchased, the cost estimate will be 10% too high. If 100 parts are purchased, the estimate will still be 10% too high. It can be seen that although the cost would change proportionally with the number of spares purchased, the error factor would remain unchanged at 10%. As previously mentioned, C_F is only dependent on C_A and C_B in that they determine the number of spares required. If it turns out that cost is proportional to the number of spares required, then the error associated with cost would not be affected by the number of spares, and therefore would also not be affected by C_A and C_B. To summarize, if cost is proportional to the number of spares, then C_F can be considered independent of C_A and $C_{B;}$, therefore, it can also be calculated separately.

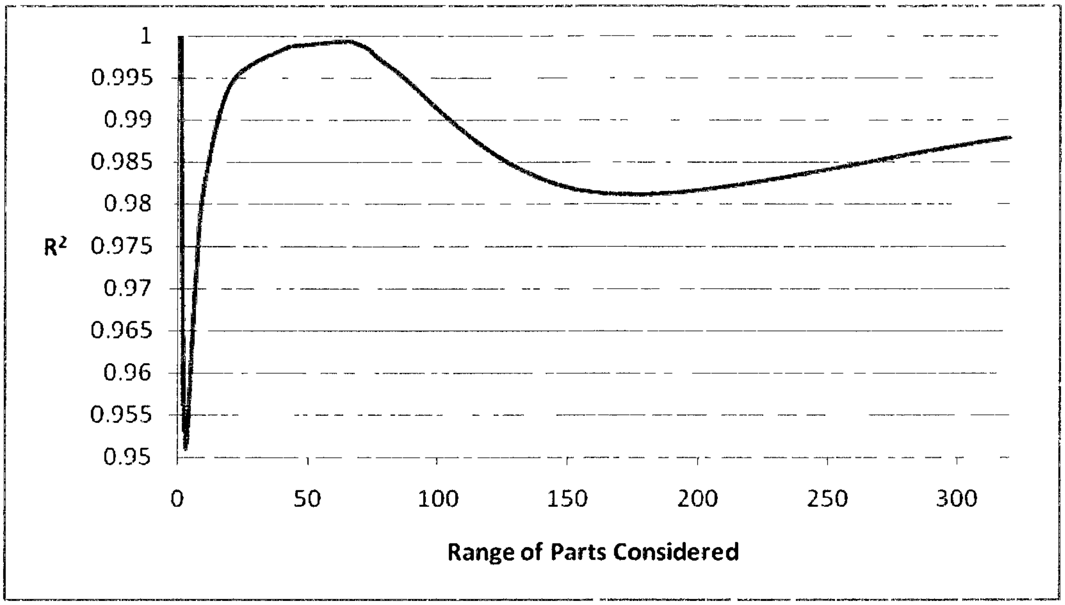
The real cost for a part, however, will not be perfectly proportional because it will feature considerations such as the following scenario. Purchasing a part costs $1000 per part. Each order placed also costs $1000 for someone to take the time to place the order, receive it, store it, etc. Alternately, an aftermarket contract can be set up with the vendor for a cost of $20k. Once that contract is in place, the cost to acquire a new part would be $700.

In this typical example, the cost is not proportional to the number of parts. However, it is quite close to being proportional. The cost, based on the number of spares required, is shown in Graph 1 below. A line showing the best-fit proportional estimate from 0 to 100 parts is also included on the graph, which fits with an R^2 value of over 0.99, making it quite accurate. In other words, while the cost may not actually be perfectly proportional to the number of spares, making that assumption still results in an R^2 of 0.99.



Graph 1 - Demonstration of Proportional Relationship Between Cost and Parts Required for P=0 to 100

The value of R^2 actually depends on the range over which the number of parts are considered. The best fit line while considering the range from 0 to 50 parts is not the same as considering the range from 0 to 100 parts. Graph 2 shows the relationship between R^2 and the range of parts considered for the given example.



Graph 2 - R^2 Value of Treating Cost as Proportional to Parts Needed

The R^2 value is shown to be consistently high, never dropping below 0.95. In fact, for any realistic range where a system includes more than 10 parts, it never drops below 0.98, demonstrating that modeling the cost as being proportional to the number of spares required is quite accurate. These charts are only for the specific scenario outlined above and scenarios with smaller R^2 values could surely be found, but the general point stands that the cost is very close to being proportional to the number of spares required. Since a proportional cost means an independent C_F, it follows that C_F is only very weakly dependent on C_B and C_C.

This is a very important result which will be relied upon to quantify the results of this research. It is possible to simulate the portion of the proposed method impacted by C_B and C_C. It is also possible to separately simulate the results of the portion of the method which is impacted by C_F. If it can be shown that C_F is only very weakly dependent on C_B and C_C, then the overall performance can be closely approximated by combining the results of these simulations as if C_F were in fact independent. As a result, there will be a known imperfection with the final results, but it should be extremely close to accurate as long as it can be demonstrated that solution costs are essentially proportional to the number of spares needed. This is a requirement that will be demonstrated to be true at a later point in this dissertation. For now, the research will proceed under the assumption that the simulations can be combined because C_F is essentially independent of C_A and C_B. Later it will be verified that this was a good assumption by using data from simulations to show that cost is in fact proportional to the number of spares needed.

The method that will be used to combine these sources of imperfect correlation and determine that the FROST Method is in fact an improvement over current methods is as follows.

$$Reduction\ in\ Cost = 1 - \frac{Cost_{proposed}}{Cost_{current}} \tag{2.7}$$

The FROST Method will reduce the cost of supporting a system compared to current methods by this amount. These costs will be based on the expected value of scenarios where both the proposed and the current methods are used to choose the better of two schedules. The equation for that cost is

$$Cost = C \cdot S + C \cdot (1 + zD) \cdot (1 - S) \tag{2.8}$$

where C represents the cost of the better schedule, S represents the success rate of choosing the better schedule, D is the difference between schedules, and z represents the spare factor. The spare factor is how many spare parts are actually needed for each part that fails. For example, if 100 parts fail but 90 are regained through repairs or harvests, then the spare factor is 10% since only 10 spares will be needed to cover the 100 failures. The difference between schedules, D, represents proportional usage. In other words, if the base schedule involved using an average of 100 instances of a part in the system, a D of 10% would indicate that the second schedule used an average of 110 instances of that part.

With this in mind, the cost equation is fairly simple. The first term, CS, is the cost of the better schedule times the chance it is successfully chosen and used. For the second schedule, the chance that it is chosen and used is (1-S). The cost of that second schedule is C(1+zD). In other words, if there were 10% more usage (D=.1) and one spare was needed for every 10 failures (z=.1), there would be a requirement for 101% of the spares in inventory compared to the base schedule. This again relies on the concept that cost is proportional to the number of spares required, which will be demonstrated later. Thus, this second schedule would cost 101% as much as the base schedule, or 1.01 C.

To differentiate between the proposed method and the current method, the subscripts p and c will be used, respectively. The FROST Method's reduction in cost, then, is

$$Reduction\ in\ Cost = 1 - \frac{C_p \cdot S_p + C_p \cdot (1 + zD) \cdot (1 - S_p)}{C_c \cdot S_c + C_c \cdot (1 + zD) \cdot (1 - S_c)} \tag{2.9}$$

There is now an equation in place which can be used to show that the FROST Method is, as claimed in the title, an improvement over the current way of doing things. If this turns out to be a positive value, it has in fact improved upon the way of doing things.

What remains is to detail the method, determine values for these variables, and demonstrate the equation above in different situations.

2.3 Review

At this stage it would be helpful to perform a quick review of what has been laid out so far and what remains to be accomplished. The following equation has been arrived at to test the effectiveness of the FROST Method.

$$Reduction\ in\ Cost = 1 - \frac{C_p \cdot S_p + C_p \cdot (1 + zD) \cdot (1 - S_p)}{C_c \cdot S_c + C_c \cdot (1 + zD) \cdot (1 - S_c)} \tag{ 2.10}$$

For this equation to be accurate, the following conditions need to be true.

1. The cost of solutions needs to be proportional to the number of spares required.
2. The determination of S, the success rate of choosing the better schedule, must incorporate the errors and randomness which cause C_B and C_C to be imperfect. This means it must include the effects of bad data due to the use of distributed inputs, as well as the fact that MTBFs are imperfect predictors.
3. The determination of C, the cost of solving a part, must incorporate the errors and randomness which cause C_F to be imperfect. This means it must include the effects of the Cost Prediction module sometime picking the incorrect solution.

If it can be demonstrated that these three conditions have been met while showing a significant positive value for reduction in cost, the FROST Method will have successfully been shown to be an improvement.

In the following section, the method will be laid out in detail. Then values for S will be calculated while meeting Condition 2. Next, values for C will be calculated while meeting Condition 3. After that, data will be analyzed to show that Condition 1 is true. Finally, the results will be combined into the reduction in cost equation, demonstrating the success and validity of the FROST Method.

CHAPTER 3

DETAIL DESIGN AND THEORY

The key variable in this process is inventory. If positive inventory exists throughout a prediction, the worst-case scenario is that a part fails and is fairly quickly replaced by a spare. If negative inventory exists at any point during a prediction, however, it means that there was a need to draw from inventory that did not exist. This is a serious problem as it means a part broke and was unable to be replaced. Either a redundant piece of a system is now operating with reduced redundancy, or a non-redundant piece has broken and the system is down.

Predicting future inventory is a fairly simple concept; take current inventory, subtract inventory out, and add inventory in. Inventory out is simply the number of failed parts. When a part fails, a replacement is drawn from inventory. Inventory in is a combination of replacements and harvests. Replacements include any way that a failed part could be replaced, such as through purchasing an additional part or repairing the failed part. Harvests are when additional inventory becomes available by removing parts from systems which are no longer needed. These concepts lead to four variables.

Before discussing these four variables further, notation needs to be discussed. Most variables will have subscripts involving some combination of the letters p, L, E, and t. Each of these subscripts represents a single entry in a corresponding index. Most variables used in this model are simple arrays, and their dimensionality can be determined by counting the number of indexes referenced. For example, $A_{p,t}$ is a two-dimensional array because it references two indexes, p and t. The four indexes are as follows.

p = Part Index

This represents a particular entry in a hypothetical one-dimensional array of all parts. It represents a specific part being analyzed and is generally used as an index for other arrays. For example, G_p represents the value of array G for a particular part, p.

t = Time Index

This represents a particular time in a hypothetical one-dimensional array of time and is used as an index for other arrays. For example, $F_{p,t}$ represents the number of failures that occurred for part p during month t.

E = Equipment Index

This represents a particular entry in a hypothetical one-dimensional array of all equipment. It is used as an index for other arrays. For example, $U_{E,L}$ represents the Usage Factor for a specific piece of equipment in a specific building.

L = Location Index

This represents a particular entry in a hypothetical one-dimensional array of all locations where powered equipment might be located, such as buildings or ships. It is used as an index for other arrays. For example, $U_{E,L}$ represents the Usage Factor for a specific piece of equipment in a specific building.

With that subscript notation now understood, the four variables discussed so far are represented as follows.

$I_{p,t}$ = Inventory Array

This is a two-dimensional array indicating the amount of spare inventory for part p that will be available during time t.

$F_{p,t}$ = Failure Array

This is a two-dimensional array indicating the number of failures for part p that will occur during time t.

$A_{p,t}$ = Replacement Array

This is a two-dimensional array indicating the number of failures for part p that will be replaced, either through repair or purchase, during time t.

$H_{p,t}$ = Harvest Array

This is a two-dimensional array indicating the quantity of part p successfully harvested and returned to inventory in working order during time t. This opportunity occurs when a piece of equipment is removed from a location.

This makes the equation for inventory of part p during time t fairly straightforward. Inventory at any time is equal to previous inventory, minus any loss in inventory, plus any gain in inventory.

$$I_{p,t} = I_{p,t-1} - F_{p,t-1} + A_{p,t-1} + H_{p,t-1} \tag{3.1}$$

The next step is to analyze the individual terms to come up with a way to predict their future values. First the failure term will be examined. When dealing with failures, "the negative exponential distribution applies whenever the probability of failure in a small time interval practically does not depend on the age of the component"(Todinov, 2005). In other words, if the failure rate for a part is independent of how old the part is, its failure rate will be a negative exponential distribution. Electronic components generally exhibit this age-independent property and making this assumption is standard process, as shown by its use throughout military handbooks (DoD, 1995). It is, however, not a perfect assumption. For example, batteries age with use and eventually "fail" by not holding enough charge to be useful. Electronics with mechanical components, such as traditional hard disk drives, wear with age. Electronics that do not involve batteries or moving parts, however, exhibit failure rates that are essentially independent of age (Wilkins, 2002). The method that will be presented is tailored for these sorts of electronics, so it should be noted that the steps that follow are inaccurate for age-dependent parts. In section nine, additional information is provided about how to use the FROST Method for age-dependent parts. The method that will be demonstrated and the results achieved from this point forward, however, will be calculated under the assumption that this method is to be applied to age-independent parts and can therefore use a negative exponential distribution.

The knowledge that electronics can be dealt with using a negative exponential distributions is important because "if the time between failures of a repairable device follows the negative exponential distribution with mean time to failure θ..., the probability density of the time to the *k*th failure is given by the gamma distribution"(Todinov, 2005). This means the gamma distribution can be used to find the probability of a certain number of failures occurring by a certain time. However, this is the reverse of what is required. Instead of inputting a number of failures and getting a probability, a function is needed where a probability can be input in order to get a number of failures.

What is needed is the Gamma Inverse distribution. It is important to differentiate this from the Inverse Gamma Distribution, which is a related but separate distribution. The Gamma Inverse is arrived at by solving the Gamma Distribution for the number of failures. There is no closed-form solution for this inverse; instead, an iterative technique must be used where different numbers of failures are plugged into the Gamma distribution until one is found which results in the desired probability.

$$\Gamma inv(\alpha_p, K, 1) = solve\ [Gamma(\alpha_p, x) = K\ ]\ for\ x \tag{3.2}$$

where x is the number of failures for a part, K is the probability that there will be x failures or less, and α_p is the usage-to-failure-rate ratio for part p.

K = Confidence Level

This represents the confidence level at which analyses are being performed. A value of 0.5 would mean results are equally likely to be better or worse than those output. A value of 0.95 would mean results have a 5% chance of being worse than outputted.

For Gamma(α_p,x), the two-parameter probability density function is

$$\frac{x^{\alpha_p-1}e^{-x}}{\Gamma(\alpha_p)} \tag{3.3}$$

(Weisstein, Gamma Distribution, 2011), and the cumulative distribution function is

$$\frac{\gamma(\alpha_p,x)}{\Gamma(\alpha_p)} \tag{3.4}$$

(Weisstein, Regularized Gamma Function, 2011) where $\gamma(\alpha_p,x)$is the Lower Incomplete Gamma Function (Weisstein, Incomplete Gamma Function, 2011) defined as

$$\gamma(\alpha_p, x) = \int_0^x t^{\alpha_p - 1} e^{-t} dt \tag{3.5}$$

So where does this usage-to-failure-rate ratio come from for the Gamma Inverse equation? It is a fairly simple concept. If a part is expected to fail every two years, a system contains five instances of that part, and the system will be used for three years , the ratio would be 7.5. This is arrived at by dividing usage (5 parts x 3 years) by failure rate (2 part years). The failure rate is simply the Mean-Time-Between-Failures (MTBF).

Θ_p = MTBF

This represents the mean time between failures for part p.

Obtaining θ_p will be discussed in more depth later. This leaves the question of where the usage rate will come from. This depends on how parts will be tracked and organized.

Systems can be thought of as being composed of pieces of equipment. A piece of equipment might be a server cabinet, a workstation, etc. Inside of these pieces of equipment are electronics such as processors and network cards. These systems and their equipment are put at different locations, such as a hospital, office building, or ship depending on the circumstance.

These four “levels” (part, equipment, system, locations) will be tracked as a three-level hierarchy with one additional level of metadata used for information filtering. The three levels of the hierarchy are:

Part > Equipment > Location

Note that this hierarchy corresponds to the subscript notation, represented by indexes p, E, and L. In addition, whenever there is an Equipment-to-Location relationship, it will be noted what system that piece of equipment is a member of. For example, there might be a processor (part) inside of a computer (equipment) inside of a classroom (location), and that computer is considered part of the school's classroom computer network (system).

The relationships in this hierarchy can be thought of as arrays of quantities. For example, there is a certain quantity of all possible parts in each piece of equipment. This quantity of parts will be denoted as Q(p). There is also a quantity of equipment at each location. This quantity of equipment will be denoted as Q(E). This introduces two new variables

$Q_{p,E}(p)$ = Quantity of Parts per Equipment

This represents the quantity of part p which exists in each instance of equipment E.

$Q_{E,L,t}(E)$ = Quantity of Equipment per Location

This represents the quantity of equipment E at location L during time t.

By combining these variables, the quantity of parts at a given location can be determined at any given time. For example, the total quantity of part p at location L during time t would be

$$\sum_{E} Q_{p,E}(p) \cdot Q_{E,L,t}(E) \tag{3.6}$$

The total actual parts in use at all locations will be denoted as C(a). This requires indexes for the part p being calculated and the time t being considered. It is found by taking the above equation and summing it for all locations.

$C_{p,t}$(a) = Actual Parts in Service

This is a two-dimensional array indicating the total parts of type p being considered at time t. This is used for accounting and harvesting but not for calculating failures.

$$C_{p,t}(a) = \sum_{L}\sum_{E} Q_{p,E}(p) \cdot Q_{E,L,t}(E) \tag{3.7}$$

This is very close to providing what is needed to find the total usage of a part in all locations. In fact, all that would be required is to multiply this by Δt to get the total usage of the part during any time period t. However, the time the part was sitting in a piece of equipment is not what is desired. Instead, what is needed is the time the part is in use, which is a subtle difference. Systems often do not manage to be powered on for 24 hours a day; there is a utilization factor which is a fraction of the time that a piece of equipment, and therefore the parts inside that equipment, is powered on and in use. Each piece of equipment at each location might be powered on a different fraction of the time, so this utilization factor will be denoted as $U_{E,L}$.

$U_{E,L}$ = Usage Factor

This represents the fraction of the time that a piece of equipment E is powered on and parts are actively failing, for a particular location L.

There is already a variable, C(a), which denotes the actual parts deployed. Another variable, C(l), which denotes the live parts deployed, is now needed.

$C_{p,t}(l)$ = Live Parts in Service

This is a two-dimensional array indicating the total parts of type p being considered which are actively powered on and failing at time t. This is used for calculating failures and includes adjustments for parts that may be deployed and harvestable but not powered on or failing.

$C_{p,t}(l)$ requires just a small change compared to equation 3.7, adding in utilization.

$$C_{p,t}(l) = \sum_{L}\sum_{E} Q_{p,E}(p) \cdot Q_{E,L,t}(E) \cdot U_{E,L} \tag{3.8}$$

This is the total parts used times the utilization factor. It provides the usage of the part at a particular time. Now a way is needed to convert this into failures at a particular time, which is what is required for $F_{p,t}$. It follows that

Failures at time t = (Usage at time t) x (failures per usage)

$$F_{p,t} = C_{p,t}(l) \cdot G_p \tag{3.9}$$

G_p = Failures per Usage

This is a one-dimensional array indicating the number of failures expected to occur if a single instance of part p is powered on for one time period, Δt.

To find the failures per usage, the total failures must be divided by the total usage. It has already been determined that the gamma inverse can be used to determine the total failures. Therefore the failures divided by usage can be found with the following calculation.

$$G_p = \frac{\Gamma inv(\alpha_p, K, 1)}{\sum_t C_{p,t}(l)} \tag{3.10}$$

All that is missing is the equation for α_p. As previously discussed, this is merely usage divided by failure rate. Total usage can be found by multiplying the parts in use times the time used and summing for all time. Dividing this by the MTBF provides the following equation.

$$\alpha_p = \sum_t \left(C_{p,t}(l) \cdot \Delta t / \theta_p\right) \tag{3.11}$$

The next part of the inventory equation to tackle is harvesting. When a piece of equipment is removed from a location, there is an opportunity to take the parts in that piece of equipment and add them back to inventory. Additionally, there can actually be equipment with a negative quantity of parts. This equipment would be a logistical fabrication that allows a piece of equipment to be noted as being added to a site which actually *removes* parts. These situations can be incorporated into a harvest variable as follows.

$$H_{p,t} = \frac{R_p(h)}{2} \sum_L \sum_E \left[\left(\begin{array}{l} \left|Q_{p,E}(p) \cdot \left(Q_{E,L,t}(E) - Q_{E,L,t-1}(E)\right)\right| \\ -Q_{p,E}(p) \cdot \left(Q_{E,L,t}(E) - Q_{E,L,t-1}(E)\right) \end{array} \right) \cdot J_{p,L,E} \right] \tag{3.12}$$

At the heart of this equation is the combination of $Q_{p,E}(p) \cdot (Q_{E,L,t}(E) - Q_{E,L,t-1}(E))$, which appears twice. This represents the change in parts at a location, which can be thought of as the quantity of parts per equipment times the change in equipment, or $Q(p) \cdot \Delta Q(E)$. This, by itself, correctly determines the change in parts. However, it needs to be slightly more complicated because there is a need to identify the harvesting opportunities mentioned above but not identify non-harvest opportunities. There are four possible situations for $Q(p) \cdot \Delta Q(E)$, shown in Table 3.

Situation	$Q(p)$	$\Delta Q(E)$	$Q(p) \cdot \Delta Q(E)$	Desired Outcome
Eqpt that removes parts is added	-X	+Y	-XY	Harvest XY parts
Eqpt that removes parts is removed	-X	-Y	XY	No Harvest
Eqpt that adds parts is added	+X	+Y	XY	No harvest
Eqpt that adds parts is removed	+X	-Y	-XY	Harvest XY parts

Table 4 - Situation vs. Desired Outcome for Harvesting

To turn $Q(p) \cdot \Delta Q(E)$ into the Desired Outcomes shown in Table 3, the following math is used: $\frac{1}{2} [\mid Q(p) \cdot \Delta Q(E) \mid - Q(p) \cdot \Delta Q(E)]$. This accounts for all of equation 3.12 except the variables $R_p(h)$ and $J_{p,L,E}$.

$J_{p,L,E}$ = Harvest Flag

This represents whether a particular part p is able to be harvested out of equipment E at location L. It is true or false and assigned a value of 1 or 0.

$R_p(h)$ = Harvest Rate

This represents the fraction of parts that are successfully harvested from a system and returned to inventory in working order for each instance of part p that is removed from a location.

The Harvest Flag allows for a piece of equipment to not be harvestable. Without it, equation 3.12 would automatically determine there was an opportunity to harvest every time a piece of equipment is removed from a site. It is possible, however, that equipment can be removed from a site and be repurposed so that no parts can be harvested out of it. This flag allows that consideration to be included in the math. The Harvest Rate allows for the possibility that sometimes parts can be broken or lost during the harvesting process, so a fraction less than 100% of harvested parts will actually end up being returned to inventory.

The last part of the inventory equation is to consider replacements. Replacements come in two forms: repairs and buys. In other words, when a part fails, it could be replaced either through repairing (supporting) the broken part or through buying a new one. This would occur whenever a part fails, with some delay. The equation for this is

$$A_{p,t} = F_{p,t-D_p(s)} \cdot S_{p,t-D_p(s)} + F_{p,t-D_p(b)} \cdot B_{p,t-D_p(b)} \qquad (3.13)$$

$D_p(s)$ = Support Delay

This represents the delay that occurs between a part p failing and that part being returned to inventory after being repaired.

D_p(b) = Buy Delay

This represents the delay that occurs between a part p failing and a new part being purchased as a replacement and put into inventory.

$S_{p,t}$ = Support Array

This is a two-dimensional array indicating the rate at which failed parts p are repaired at time t.

$B_{p,t}$ = Buy Array

This is a two-dimensional array indicating the rate at which failed parts p are replaced by new purchases at time t.

So, whenever a part fails, after a delay a certain portion of parts are returned to inventory through repairs and through buys. The next question is how to determine what those portions are. The first issue to consider is that parts cannot always be purchased and repaired. Eventually each of these options end. The date when a part is no longer available to be bought will be noted as M_p(b), and the date when a part is no longer supported and cannot be repaired will be noted as M_p(s).

$M_p(s)$ = End of Support

This represents the date when repairs are no long available for part p.

$M_p(b)$ = End of Buy

This represents the date when purchasing additional parts is no longer an option for part p.

Depending on the situation, there could be different rules for whether a purchase or a repair is preferred. This method can easily handle any of those rules, but for purposes of this research the assumption will be made that repairs are a default option with services already paid for, so if a part fails someone will try to repair it. On the other hand, the assumption will be made that buying new parts is not automatic, and this only happens if the method notifies someone they need to take action and make a purchase. If someone were to try to use this method on a system where purchasing actually is automatic, or repairing is not, this can easily be tweaked.

$$B_{p,t-D_p(b)} = 0 \tag{3.14}$$

The total buys being assumed will be zero. This does not mean buys will not happen; it simply means they are not being considered as a default, automatic option, and action and cost will be associated with making a buy. This will be discussed further in a later section.

Now that the decision to make repairs automatic has been considered, the method needs to allow for the possibility that a specific part might not be repairable. For example, in many classified systems hard drives are not considered repairable because the cost associated with securely transferring a classified hard drive to a repair facility can exceed the cost of a new hard drive. This process will be noted with flag V_p. It also needs to be considered that not every part is successfully repaired.

v_p = Repair Flag

This is a variable representing whether or not part p is repairable. 1 means repairable, 0 means not repairable.

$R_p(s)$ = Support Rate

This represents the fraction of parts that are successfully repaired for failures of part p while repair is still available.

Lastly, this support rate is only valid up until the End of Support date. Putting this all together, the equation for overall support for part p during time t is shown.

$$S_{p,t} = \begin{cases} R_p(s) \cdot V_p, & t < \mathrm{M_p(s)} \\ 0, & t \geq \mathrm{M_p(s)} \end{cases} \tag{3.15}$$

This completes the basic math behind calculating future inventory. It details most of the operation of the Prediction of Health module. All that remains is to make this operate as part of a Monte Carlo process by providing distributions to sample from and then determining at what time t the inventory equation $I_{p,t}$ first drops below zero for each part p. When it drops below zero, this indicates negative inventory and a part that needs a solution. Therefore, this triggers the Sustainment Effort module.

The equations presented are much easier to read when left independent, but, for reference, when combined the following becomes the equation for future inventory.

$$I_{p,t} = I_{p,t-1} - \left[\sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t-1}(E) \cdot U_{E,L} \right.$$

$$\left. \cdot \frac{\Gamma inv(\sum_t (C_{p,t} \sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t}(E) \cdot U_{E,L}) \cdot \Delta t/\theta_p, K, 1)}{\sum_t \sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t}(E) \cdot U_{E,L}} \right]$$

$$+ \left[\sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t-D_p(s)}(E) \cdot U_{E,L} \right.$$

$$\left. \cdot \frac{\Gamma inv(\sum_t (C_{p,t} \sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t}(E) \cdot U_{E,L}) \cdot \Delta t/\theta_p, K, 1)}{\sum_t \sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t}(E) \cdot U_{E,L}} \right]$$

$$\cdot \begin{bmatrix} R_p(s) \cdot V_p, & t - D_p(s) < \mathrm{M_p(s)} \\ 0, & t - D_p(s) \geq \mathrm{M_p(s)} \end{bmatrix}$$

$$+ \left[\frac{R_p(h)}{2} \sum_L \sum_E \left[\left(\left| Q_{p,E}(p) \cdot \left(Q_{E,L,t-1}(E) - Q_{E,L,t-2}(E) \right) \right| - Q_{p,E}(p) \right. \right. \right.$$

$$\left. \left. \left. \cdot \left(Q_{E,L,t-1}(E) - Q_{E,L,t-2}(E) \right) \right) \cdot J_{p,L,E} \right] \right]$$

Where

$$\Gamma inv(\alpha_p, K, 1) = solve \left[Gamma(\alpha_p, x) = K \right] for\ x \tag{3.16}$$

3.1 Inputs

The following is a list of inputs that need to be provided in order to solve the equations listed in the previous section. Note that all equations and descriptions are summarized and available in the appendix.

System and Model-level

K = Confidence Level

t = Time Index

Equipment-Level

$Q_{E,L,t}(E)$ = Quantity of Equipment per Location

$U_{E,L}$ = Usage Factor

Part-level

$D_p(s)$ = Repair Delay

$J_{p,L,E}$ = Harvest Flag

$M_p(b)$ = End of Buy

$M_p(s)$ = End of Support

$R_p(h)$ = Harvest Rate

$R_p(s)$ = Support Rate

$Q_{p,E}(p)$ = Quantity of Parts per Equipment

V_p = Repair Flag

Θ_p = MTBF

CHAPTER 4

DISTRIBUTIONS

The equations outlined in the previous section require knowledge of values for end of production dates, failure rates, etc. As previously stated, these data sources are often inaccurate. The solution is to convert these inputs into distributions which logically account for the inherent inaccuracy. For example, if a vendor says they will produce a part for two more years, there is a need to have some idea of the likelihood they actually produce it for only one year or for three years. A properly formulated distribution will account for these possibilities.

To come up with these distributions, known mathematical relationships will be used as well as data mining. With data mining, there are assumptions to keep in mind which were made and affect the reliability.

Having a somewhat limited number of data points, I chose to group all data together without separating based on any factors. Accuracy could potentially be improved by acquiring a larger data set and breaking it down into multiple distributions.

For example, 180 different data points were collected featuring a variety of types of electronics from a variety of vendors. This was used to create a single distribution which shows the likelihood of a generic vendor producing a part longer or shorter than claimed. The reality is that there are probably some vendors who always stick to their claims, others who tend to stop production early, and others who tend to stop production late. Thus, if enough data were available, it would probably be appropriate to create one distribution for parts from Motorola, another for parts from Intel, etc. This would improve the correlation for the Distributed Inputs module and further improve the performance of the FROST Method. For the purposes of validating this method, the generic distributions will be used.

4.1 MTBFs

One required input is the MTBF for a part. Typically this is estimated by simply dividing the hours of use by the number of failures witnessed. However, this is merely an estimate. Just because only one part has failed in the last thousand hours does not mean that rate will necessarily continue in the future. There is uncertainty associated with using this method to estimate a part's MTBF. As would be expected, an MTBF based on a large number of failures, or samples, is more certain than one based on a small number. Luckily, there is a known mathematical relationship to quantify this uncertainty for Mean Time to Failures (MTTFs). Technically, the difference between an MTBF and a MTTF is that MTBFs factor in the actual time it takes to replace a failed part and MTTFs do not. In normal practice, however, the replacement time is assumed to be approximately zero, and the terms are used interchangeably.

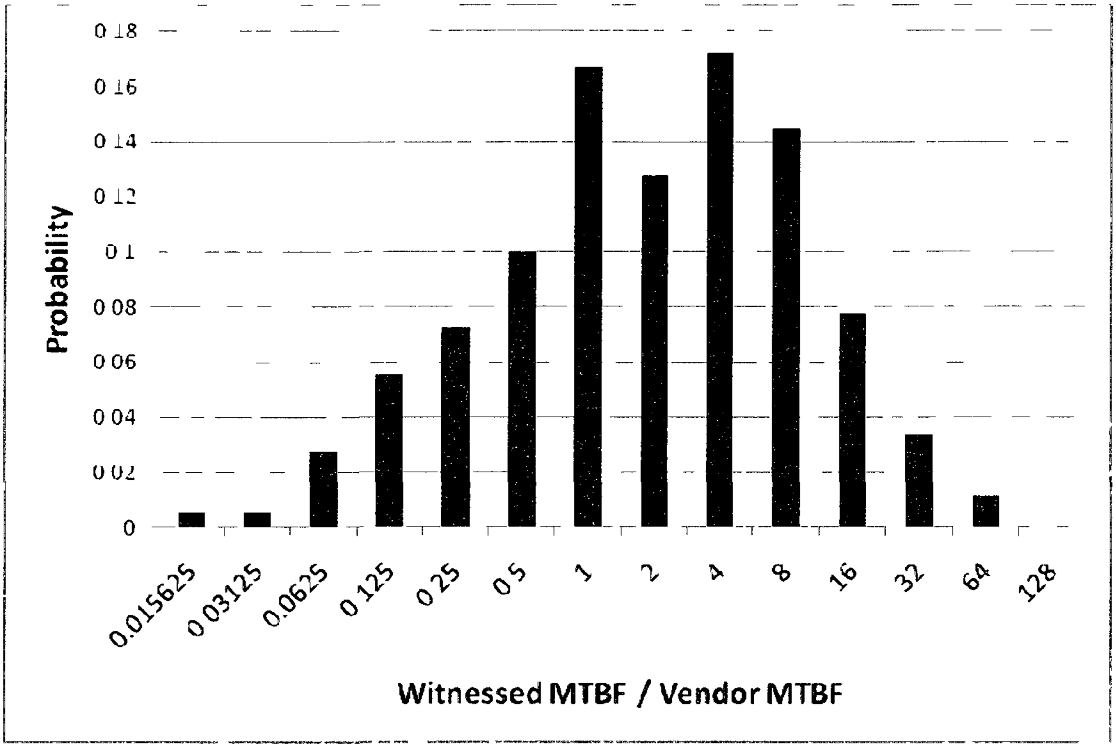
The relationship between the true MTTF's boundary and available inputs is

$$\theta = \frac{2T}{\chi^2_{\alpha,2k}} \tag{4.1}$$

where T is the time of use witnessed for the part, k is the number of failures witnessed for the part, and the denominator represents a Chi-Square distribution for probability α with 2k degrees of freedom (Todinov, 2005). By inserting a known value for T and k for any part, a distribution can be found for true MTBF. Randomly and uniformly selecting a value for α from 0 to 1 and substituting it into this equation will provide a correctly distributed random sample for a possible long-term MTBF value for a part.

While this is the preferred way to create a sample set of MTBFs for the Monte Carlo process, sometimes there is not enough history to use this method. Often the only source of MTBF data is what the vendor claims on spec sheets.

To determine the relationship between vendor claims and reality, the results of several thousand vendor surveys were obtained which included data on estimated MTBFs. These surveys were then matched up to several thousand parts' worth of historical failure data. The resulting cross-sample contained 180 data points. A simple histogram of the data is shown in Graph 3.



Graph 3 - Witnessed vs. Vendor Claimed MTBF

The standard business practice of modern electronics companies is evident in this graph. Instead of risking unsatisfied customers complaining about parts not reaching the claimed MTBF, the vendors are claiming their parts will last, on average, only 19% as long as they actually do. At first this might seem like a purely positive thing; after all, who would be upset by a part turning out to be even higher quality than expected? However, from a predictive view this immediately demonstrates a huge problem in models that take MTBF claims at face value; doing so could cause a system to procure five times as many parts as necessary.

Data fitting software was used to match a distribution to this data. The best fit was a three-parameter Burr distribution. The parameters for this distribution are continuous shape parameters k and α, and continuous scale parameter β. The PDF for this distribution is

$$f(x) = \frac{\alpha k \left(\frac{x}{\beta}\right)^{\alpha-1}}{\beta\left(1+\left(\frac{x}{\beta}\right)^{\alpha}\right)^{k+1}} \tag{4.2}$$

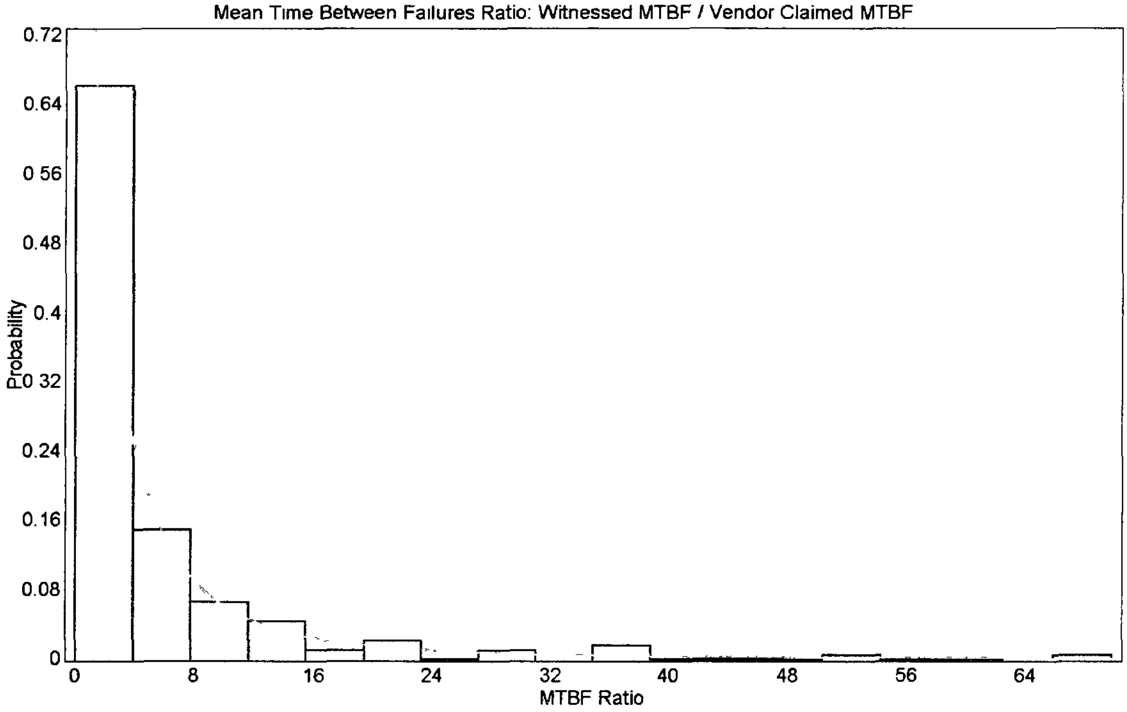
All PDFs presented in this section were taken from the descriptions provided along with the outputs of MathWave EasyFit, which is the software used for distribution fitting. The best-fit parameter values were

k=2.4483

α=0.85988

β=7.6449

which resulted in Kolmogorov-Smirnov test score of 0.035 (lower is better) and a Chi-Squared statistic of 1.7512, which is quite good. The resulting fit is shown in Graph 4.



Graph 4 - Distribution Fit for Vendor-Claimed MTBFs

4.2 End of Production

The next input to tackle is End of Production dates claimed by vendors. The results of 8,744 surveys were collected for a representative sample of electronics used in Navy systems. Of these, the surveys where the vendors responded with an End of Production date that was earlier than the date of the survey itself were isolated. The reason for this is that these surveys represent fact instead of estimates; these were examples of the vendors verifying that EOP has in fact already occurred, and confirming what date it occurred upon. These were then matched to earlier surveys for the same parts, back when the estimated End of Production was still listed as being in the future.

The result of this is that for every match, there is an example of what the vendor initially claimed the End of Production would be while the part was still in production, and then what the End of Production actually ended up being. This resulted in 395 sets of data.

For each match the time between the vendor survey and the actual End of Production was determined. The time between the vendor survey and the claimed End of Production was also determined. Then, the ratio of these two values was found. For example, if the vendor claimed there were 4 months of production remaining, and there actually ended up being 7 months, this ratio would be 7/4, meaning they produced the part for 7/4ths as long as claimed. This method allows time to automatically be included in the factor, as the impact of this 7/4ths reduces over time. 7/4ths of the remaining time is a huge variation when there are many years left, but a much smaller variation when there are only days left until EOP. This accounts for the fact that uncertainty should reduce as it gets closer to EOP and the variables that control whether or not production stops become more certain. In 218 out of 395 data sets, this ratio turned out to be exactly one. This means the vendor's claim was accurate to the day. In 6 out of 395 data sets there was a negative ratio, indicating that at one point a vendor claimed the part was still in production when in fact it was not. The rest were scattered.

There are likely two methods behind these production processes. The first is that the vendors eventually end production for market or inventory reasons and are providing estimates based on when they think their inventory or market situation will be such that they must end production. The other method is a defined business process that says "we will produce this part for x number of years and then end production that day regardless of market situation." These two processes will fit different distributions, so attempting to match a single distribution to this data would be taking an incorrect path. Luckily the distribution for the latter process is clear; 100% of the parts in this sample will occur exactly at a ratio of one. And, in fact, it can be fairly safely assumed that all of the values of exactly one were a result of this method, because the odds of a vendor meeting their estimate exactly to the day based on random chance rather than policy is exceptionally small.

Rather than attempt to fit a single distribution to all 395 ratios, they will be broken out into 3 sets. First, there is a 6 out of 395 chance that a part will be out of production regardless of what the vendor claims. Second, there is a 218 out of 395 chance that the vendor will produce a part exactly as long as claimed. Last, the remaining 171 out of 395 chance fits the following distribution, which was found by fitting only those 171 ratios which were positive but not exactly one.

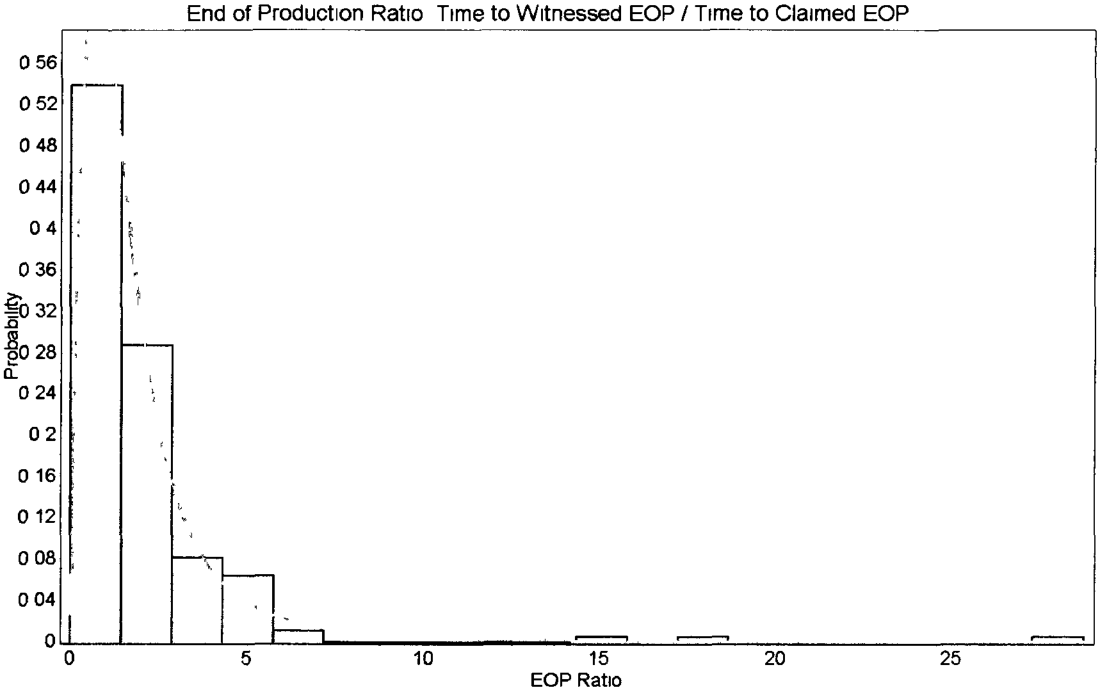
The best-fit distribution for this set was again a three-parameter Burr. The best-fit parameter values were

$k=1.3081$

$\alpha=1.7704$

$\beta=1.6636$

which resulted in Kolmogorov-Smirnov test score of 0.041 and a Chi-Squared statistic of 2.008. The resulting fit is shown in Graph 5.



Graph 5- Distribution Fit for End of Production

4.3 End of Support

This process was then repeated for End of Support dates claimed by the vendor, this time with a slightly smaller data set featuring 133 matches. In 6 out of 133 the End of Support is in the past regardless of vendor claim. In 64 out of 133 the End of Support is exactly as claimed. The remaining 63 out of 133 were fit to a distribution.

The found best-fit was a General Extreme Value distribution. This is a three-parameter distribution with continuous shape parameter k, continuous scale parameter σ, and continuous location parameter μ. This has a PDF of

$$f(x) = \begin{cases} \frac{1}{\sigma}\exp\left(-(1+kz)^{-\frac{1}{k}}\right)(1+kz)^{-1-1/k}\ when\ k \neq 0 \\ \frac{1}{\sigma}\exp(-z-\exp(-z))\ when\ k = 0 \end{cases} \tag{4.3}$$

where z=(x-μ)/σ.

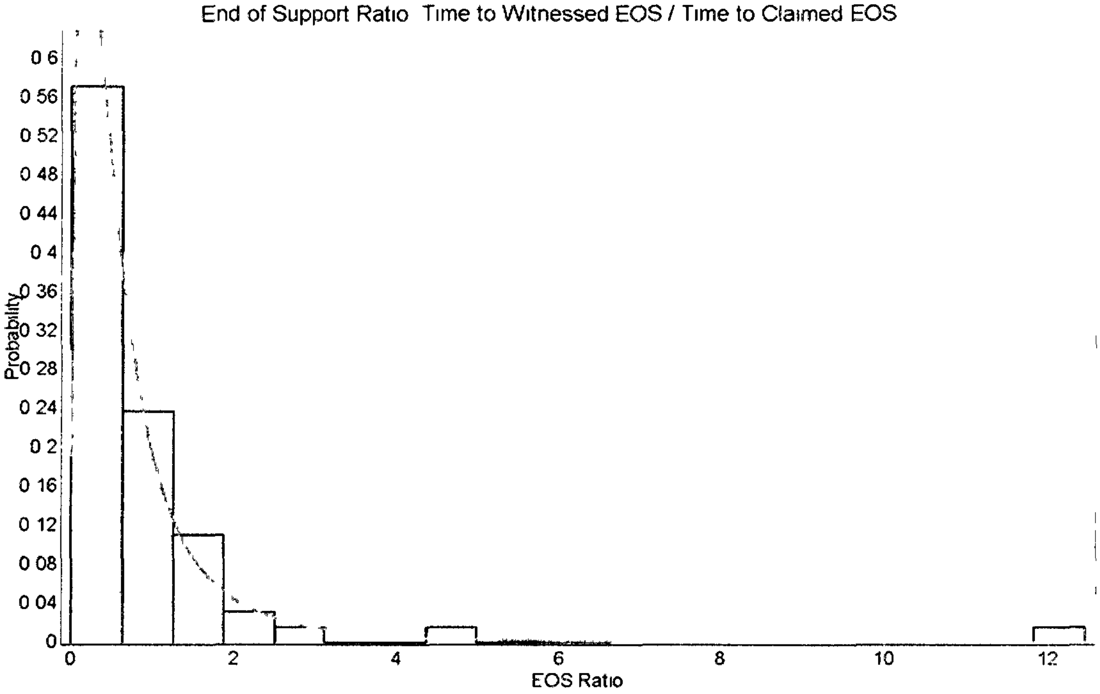
The best-fit parameter values were

k=0.53333

σ=0.32944

μ=0.34143

which result in a Kolmogorov-Smirnov score of 0.063 and a Chi-Squared statistic of 1.2557. The best-fit is shown in Graph 6.



Graph 6- Distribution Fit for End of Support

4.4 Repair Rates

Last, a distribution is needed which shows the expected repair rate for an electronic part when no part-specific information is available. 589 data points were gathered based on attempts at repairing various electronic parts, and a data fit was performed. The best-fit was a Kumaraswamy distribution. The Kumaraswamy distribution is a four-parameter distribution. However, two of the parameters represent the boundary of possible values, so with repair rates being possible only in the range of zero to one, it reduces to a two-parameter distribution with continuous shape parameters α_1 and α_2. The PDF for the two-parameter distribution is

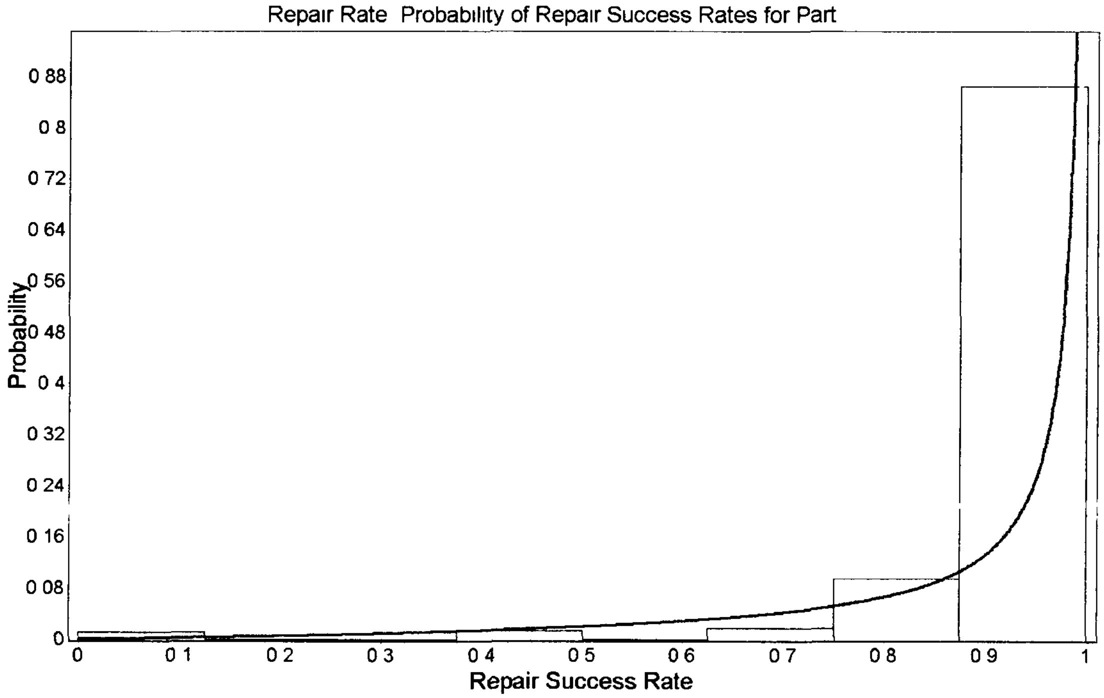
$$f(x) = \alpha_1\alpha_2 x^{\alpha_1 - 1}(1 - x^{\alpha_1})^{\alpha_2 - 1} \tag{4.4}$$

The best-fit parameter values found were

α_1=2.2636

α_2=0.14035

which resulted in a Kolmogorov-Smirnov score of 0.09451. The resulting distribution fit is shown in Graph 7.



Graph 7 - Distribution Fit for Repair Rate

These distributions are summarized and available in the appendix. With these distributions in hand, simulations can now be performed to determine the FROST Method's effectiveness.

CHAPTER 5

EFFECTIVENESS OF PREDICTING BETTER SCHEDULE

5.1 FROST Method

The next goal is to determine the success rate for choosing a better schedule with the FROST Method. This starts with equation 3.1, which was derived earlier and is shown again below for convenience.

$$I_{p,t} = I_{p,t-1} - F_{p,t-1} + A_{p,t-1} + H_{p,t-1}$$

This example is only concerned with one theoretical part instead of an array of parts, p, so the notation can be simplified by removing the factor p.

$$I_t = I_{t-1} - F_{t-1} + A_{t-1} + H_{t-1} \qquad (5.1)$$

As previously shown in equation 3.9, failures are a function of the Live Parts in Service, $C_{p,t}(l)$, and the Failures per Usage, G_p.

$$F_{p,t} = C_{p,t}(l) \cdot G_p \qquad (5.2)$$

While $C_{p,t}(l)$ is really a function of time and the number of parts in service can be in flux, for purposes of this estimation this will be simplified by replacing this with a constant, C, representing the average parts in use throughout the life of a system. Additionally, while an inverse gamma function was used in equation 3.10 to determine failures with different levels of confidence, with average confidence across an infinite amount of time the Failures per Usage converges at the inverse of Mean-Time-Between-Failure. As a result, the equation for average failures simplifies to

$$F_t = \frac{C}{\theta} \tag{5.3}$$

As previously shown in equation 3.13, reproduced below, if buys are assumed to not be a default behavior, replacements are a function of failures and repairs.

$$A_{p,t} = F_{p,t-D_p(s)} \cdot S_{p,t-D_p(s)}$$

Additionally, according to equation 3.15 the repairs are

$$S_{p,t} = \begin{cases} R_p(s) \cdot V_p, & t < \mathrm{M_p(s)} \\ 0, & t \geq \mathrm{M_p(s)} \end{cases}$$

To simplify, for determining inaccuracy the worst-case scenario will be used where the repair flag is always on, meaning v_p=1. This is worst-case because otherwise, if v_p=0, the repair rate would be multiplied by zero and therefore irrelevant. That would negate the impact of an inaccurate repair rate. Instead, that inaccuracy will be included to show the harmful impact it can have on predictions. Additionally, it will be assumed that the delay between failures and repairs, $D_p(s)$, is zero since it is not relevant to the accuracy of inputs. This simplifies the replacements equation to

$$A_t = \frac{C}{\theta} \cdot \begin{cases} R_p(s), & t < \mathrm{M_p(s)} \\ 0, & t \geq \mathrm{M_p(s)} \end{cases} \tag{5.4}$$

Substituting these equations back into the original inventory equations gives

$$I_t = I_{t-1} - \frac{C}{\theta} + \frac{C}{\theta}\left(\begin{cases} R_p(s), & t < \mathrm{M_p(s)} \\ 0, & t \geq \mathrm{M_p(s)} \end{cases} \right) + H_{t-1} \tag{5.5}$$

This time based function requires knowledge of inventory at time t-1. Instead, it would be preferable to determine inventory at time T without any dependencies on previous inventory other than knowledge of the starting inventory. This can be found by iterating this equation from t=1 to t=T, which gives the following

$$I_T = I_0 - \sum_{t=1}^{T} \frac{C}{\theta} + \sum_{t=0}^{\mathrm{M_p(s)}} \frac{C \cdot R_p(s)}{\theta} + \sum_{t=1}^{T} H_{t-1} \tag{5.6}$$

with the requirements that $M_p(s) >= 0$ and $T >= 0$. This further simplifies to

$$I_T = I_0 + \left(\mathrm{M_p}(s) \cdot R_p(s) - T\right) \cdot \tfrac{C}{\theta} + \textstyle\sum_{t=1}^{T} H_{t-1} \tag{5.7}$$

Under the previously mentioned assumptions that usage levels are stable and roughly C parts are deployed at any given time, this provides a good approximation of the inventory at time T from which to start estimating the inaccuracy introduced into the model. There are two sources of uncertainty. The first is if values turn out to be different than expected. For example, if a part were expected to fail every 1000 hours but it failed every 800 hours instead, the predictions would be off. This applies to repair rates, end of support dates, and MTBFs. When predictions are being made, there is uncertainty and inaccuracy which has been thoroughly discussed and is accounted for with distributions. However, once a prediction has been made and the real outcome has come to pass, there is no longer uncertainty. Instead, all quantities are known, and the difference between the values predicted and the actual outcome is not uncertainty, but error. These errors will be accounted for by attaching an error factor to these variables. For example, if the expected repair rate turns out to be only 90% of the actual repair rate, what was actually fed into the equation above was not $R_p(s)$ but $0.9R_p(s)$. These error factors will be designated as efr for the repair rate error factor, efs for the support date error factor, and efm for the MTBF error factor.

The second source of errors is from the distributed inputs module. Applying these distributions to estimates will introduce additional error. These distribution factors will be designated as dfr for the repair rate error factor, dfs for the support date error factor, and dfm for the MTBF error factor.

Therefore, including error the equation becomes

$$I_{T\,w/error} = I_0 + \left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs \cdot R_p(s) \cdot efr_p \cdot dfr - T\right) \cdot \frac{C}{\theta \cdot efm_p \cdot dfm} + \sum_{t=1}^{T} H_{t-1}$$

(5.8)

This shows the prediction that will be made for the inventory level for one part, with a single-run simulation. However, in actually applying the proposed method, the discussed Monte Carlo process will mean each part features many runs, not just one. In addition, a system will almost certainly be composed of multiple parts.

A further consideration is that the whole method is built around the concept of minimizing cost, not inventory. The equation above provides estimates of inventory, but not cost. For now a simple way is needed to relate this inventory requirement to expected costs. At this stage, the assumption will again be made that cost is proportional to the inventory required to support each part, and also that each part can have a different proportional cost attached to it. In other words, the need for one additional spare might only cost $20 for a network card, but $5000 for a high performance processor.

Thus, for a system with N runs per simulation, P parts, T lifetime, and a weighted cost for each part of w_p, the predicted total cost would be

$$Cost_{Prediction,A} = \frac{1}{N}\sum_{n=1}^{N}\sum_{p=1}^{P} w_p \cdot I_{T\,w/error} \tag{5.9}$$

$$Cost_{Prediction,A} = \frac{1}{N}\sum_{n=1}^{N}\sum_{p=1}^{P} w_p \left[I_0 + \left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs_A \cdot R_p(s) \cdot efr_p \cdot dfr_A - T\right) \cdot \frac{C_p}{\theta_p \cdot efm_p \cdot dfm_A} + \sum_{t=1}^{T} H_{t-1} \right] \quad (5.10)$$

When working with this equation, real data found from commercial-parts based Navy systems will be used, operating under the assumption that these systems are fairly representative of electronics systems as a whole since they are mostly composed of standard parts such as hard drives, network cards, graphics processors, memory, etc. Historical data mining was performed for several COTS electronics-based Navy systems and, based on 133 data points, it was determined that the average time until end of vendor support for a part is 1.58 years. Additionally, based on a history of 589 repair attempts, a repair success rate of 94.2% was discovered. This data comes from the same systems used to generate the input distributions, and provides values for $M_p(s)$ and $R_p(s)$ in this simulation.

Another way to think of inventory error is by breaking down a typical inventory scenario. If an assumption is made that a system would keep on hand an amount of inventory proportional with factor z to the failures they expect to need in the future, then a rough estimate of inventory on hand at any time would be $z \cdot (C \cdot T / \theta)$, with T being the amount of time the system will be fielded. Note that z is the same spare factor discussed in the validation section. This would give a calculation of z times the expected failures remaining, and many systems would keep this amount on hand. The idea is that a fraction of failures equal to (1-z) gets returned to inventory through repairs and harvests, and the rest of the failures need to be covered by spare inventory. For example, if 75% of the loss in inventory is going to be replaced through repairs and harvesting, then to maintain positive inventory a system would need 25% ($C \cdot T$ / θ) spares. This leaves an equation for actual inventory without error yet included. In this case, the z term accounts for repairs and harvests.

$$I_{Actual} = z\frac{C \cdot T}{\theta} \tag{5.11}$$

Assume that the goal is to determine the better of two schedules, A and B. A is based on the current, planned schedule for the system and so will result in the already determined actual inventory.

$$I_{Actual}[A] = z\frac{C \cdot T}{\theta} \tag{5.12}$$

Schedule B, on the other hand, will have some changes. These changes will result in a different total usage of the part in question, and therefore a different amount of failures. As a result, it will require a different number of spares to maintain supportability. To approximate this, assume that for a part under schedule B, the usage will change from C to C(1-D). Therefore, if D = 10%, this would represent a schedule change significant enough to decrease usage by 10%.

$$I_{Actual}[B] = z\frac{C\cdot(1-D)\cdot T}{\theta} \tag{5.13}$$

If the exact same approach is taken when converting predicted inventory into a cost, the actual cost for each of these schedules can be found.

$$Cost_{Actual,A} = \sum_{p=1}^{P} w_p \cdot z\frac{C_p\cdot T}{\theta_p} \tag{5.14}$$

$$Cost_{Actual,B} = \sum_{p=1}^{P} w_p \cdot z\frac{C_p\cdot(1-D_p)\cdot T}{\theta_p} \tag{5.15}$$

The better schedule will be the one whose actual cost is the least. So, Schedule A would be better if

$$Cost_{Actual,A} < Cost_{Actual,B} \tag{5.16}$$

which is

$$\sum_{p=1}^{P} w_p \cdot z\frac{C_p\cdot T}{\theta_p} < \sum_{p=1}^{P} w_p \cdot z\frac{C_p\cdot(1-D_p)\cdot T}{\theta_p} \tag{5.17}$$

and simplifies to

$$\sum_{p=1}^{P} w_p \cdot D_p \frac{C_p}{\theta_p} < 0 \tag{5.18}$$

This means the sign of the left side of the equation above will tell the better schedule. If it is negative, schedule A is better. If positive, schedule B is better. This is worth noting as that fact will be used later.

Next, a similar equation for the prediction needs to be found. Earlier in equation 5.10 the prediction for a schedule without change was shown, which has since been designated Schedule A.

$$Cost_{Prediction,A} = \frac{1}{N}\sum_{n=1}^{N}\sum_{p=1}^{P} w_p \left[I_0 + \left(\mathrm{M_p}(\mathrm{s}) \cdot R_p(s) \cdot efs_p \cdot dfs_A \cdot efr_p \cdot dfr_A - T\right) \cdot \frac{C_p}{\theta_p \cdot efm_p \cdot dfm_A} + \sum_{t=1}^{T} H_{t-1} \right] \tag{5.19}$$

To find the predicted cost for schedule B, the substitution is again performed where C becomes C(1-D).

$$Cost_{Prediction,B} = \frac{1}{N}\sum_{n=1}^{N}\sum_{p=1}^{P} w_p \left[I_0 + \left(\mathrm{M_p}(\mathrm{s}) \cdot R_p(s) \cdot efs_p \cdot dfs_A \cdot efr_p \cdot dfr_A - T\right) \cdot \frac{C_p \cdot (1-D_p)}{\theta_p \cdot efm_p \cdot dfm_A} + \sum_{t=1}^{T} H_{t-1} \right] \tag{5.20}$$

Schedule A will be selected if it is predicted to have a smaller cost than Schedule B. This occurs when

$$\sum_{n=1}^{N}\sum_{p=1}^{P} w_p \left[\left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs_A \cdot R_p(s) \cdot efr_p \cdot dfr_A - T\right) \cdot \frac{C}{\theta \cdot efm_p \cdot dfm_A}\right] < \sum_{n=1}^{N}\sum_{p=1}^{P} w_p \left[\left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs_B \cdot R_p(s) \cdot efr_p \cdot dfr_B - T\right) \cdot \frac{C \cdot (1-D)}{\theta \cdot efm_p \cdot dfm_B}\right] \quad (5.21)$$

which simplifies to

$$\sum_{n=1}^{N}\sum_{p=1}^{P} w_p \cdot \frac{C}{\theta \cdot efm_p}\left[\left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs_B \cdot R_p(s) \cdot efr_p \cdot dfr_B - T\right) \cdot \frac{(1-D)}{dfm_B} - \left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs_A \cdot R_p(s) \cdot efr_p \cdot dfr_A - T\right) \cdot \frac{1}{dfm_A}\right] > 0 \quad (5.22)$$

There is now an equation where, if it is positive, Schedule A is chosen and if it is negative, schedule B is chosen. Notice that this is the exact opposite of the result for the actual cost.

For a prediction to be correct, it has to match the result of the actual equation. In other words, the prediction is correct of it picks schedule A and schedule A is actually cheaper, or if it picks B when B is cheaper. It was previously shown that the sign of the left side of these equations determines which schedule is actually cheaper, or predicted to be cheaper. This means that whether or not the prediction was correct can be determined by dividing the left side of the equation for the prediction by the left side of the equation for the actual result.

Situation	Prediction Equation	Actual Equation	Prediction / Actual	Result
Predict A when A is better	+	-	-	Correct
Predict A when B is better	+	+	+	Wrong
Predict B when A is better	-	-	+	Wrong
Predict B when B is better	-	+	-	Correct

Table 5 - Sign of Prediction Equation vs. Result

As shown in Table 5, whenever the prediction formula divided by the actual formula is negative, the prediction chooses the correct schedule. When it is positive, it chooses the wrong schedule. Thus, the FROST Method correctly predicts the better schedule when the left side of the prediction equation divided by the left side of the actual equation is below zero.

$$\left\{\sum_{n=1}^{N}\sum_{p=1}^{P} w_p \cdot \frac{C}{\theta \cdot efm_p}\left[\left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs_B \cdot R_p(s) \cdot efr_p \cdot dfr_B - T\right) \cdot \frac{(1-D)}{dfm_B} - \left(\mathrm{M_p(s)} \cdot efs_p \cdot dfs_A \cdot R_p(s) \cdot efr_p \cdot dfr_A - T\right) \cdot \frac{1}{dfm_A}\right]\right\} / \left\{\sum_{p=1}^{P} w_p \cdot D_p \frac{C_p}{\theta_p}\right\} < 0 \tag{5.23}$$

This provides an equation which can be used to predict the success rate of the FROST Method for choosing a better schedule based on the variables N, T, D, and P.

5.2 Previous Method

Next, a similar success equation is needed for the current method of doing business so that the FROST Method can be compared against it.

For the current method,

$$I_{T\,w/error} = I_0 + T \cdot \Delta I \tag{5.24}$$

where ΔI is the historical average change in inventory for each time period Δt. As previously mentioned, this is a standard, popular method because it is very simple. It does not require tracking MTBFs, repairs, harvests, etc. It simply requires monitoring inventory levels and watching the rate at which they drop. It is as simple as saying, if inventory has dropped by 10 parts in the last 2 years, ΔI = -5 parts per year. This is multiplied by time T to determine the overall change in inventory expected.

Since equation 3.1 says that inventory can be calculated as

$$I_{p,t} = I_{p,t-1} - F_{p,t-1} + A_{p,t-1} + H_{p,t-1}$$

The change in inventory ΔI can be found as

$$\Delta I = -F_{avg} + A_{avg} + H_{avg} \tag{5.25}$$

which is equivalent to

$$\Delta I = \tfrac{1}{T}\textstyle\sum_{t=0}^{T-1} -F_t + A_t + H_t \tag{5.26}$$

Recalling equations 5.3 and 5.4 for these terms,

$$F_t = \frac{C}{\theta}$$

$$A_t = \frac{C}{\theta} \cdot \begin{cases} R_p(s), & t < \mathrm{M_p(s)} \\ 0, & t \geq \mathrm{M_p(s)} \end{cases}$$

This reduces to

$$\Delta I = -\frac{C}{\theta} + \frac{C}{\theta}\left(R_p(s) \cdot M_p(s)\right) + \frac{1}{T}\sum_{t=0}^{T-1} H_t \tag{5.27}$$

Plugging this equation back into equation 5.24 and including error terms gives

$$I_{T\,w/error} = I_0 + T\frac{C}{\theta}\frac{1}{efm}\left(R_p(s) \cdot efr \cdot M_p(s) \cdot efs - 1\right) + \sum_{t=0}^{T-1} H_t \tag{5.28}$$

Just as before, the overall system cost will be estimated by performing this prediction for P parts, each with a weighted cost proportional to the inventory required to support it.

$$Cost_{Prediction,A} = \sum_{p=1}^{P} w_p \left[I_0 + T\frac{C_p}{\theta_p}\frac{1}{efm_p}\left(R_p(s)\cdot efr_p \cdot M_p(s)\cdot efs_p - 1\right) + \sum_{t=0}^{T-1} H_t \right] \tag{5.29}$$

Next the conversion is repeated for Schedule B, substituting C(1-D) for C.

$$Cost_{Prediction,B} = \tag{5.30}$$

$$\sum_{p=1}^{P} w_p \left[I_0 + T\frac{C_p(1-D_p)}{\theta_p}\frac{1}{efm_p}\left(R_p(s)\cdot efr_p \cdot M_p(s)\cdot efs_p - 1\right) + \sum_{t=0}^{T-1} H_t \right]$$

Schedule A will be chosen when the prediction for Schedule A is cheaper than the prediction for Schedule B. This simplifies to

$$\sum_{p=1}^{P} w_p \left[-D_p \cdot T\frac{C_p}{\theta_p}\frac{1}{efm_p}\left(1 - R_p(s)\cdot efr_p \cdot M_p(s)\cdot efs_p\right) \right] > 0 \tag{5.31}$$

Performing the same logic as earlier and dividing by the equation for actual better schedule, the success equation for the current method is

$$\frac{\sum_{p=1}^{P} w_p \left[-D_p \cdot T\frac{C_p}{\theta_p}\frac{1}{efm_p}\left(1 - R_p(s)\cdot efr_p \cdot M_p(s)\cdot efs_p\right) \right]}{\left\{\sum_{p=1}^{P} w_p \cdot D_p \frac{C_p}{\theta_p}\right\}} < 0 \tag{5.32}$$

5.3 Results and Analysis

Before moving on, an analysis will be performed using real data in the two equations for success. To understand the performance of the two methods in all situations, six degrees of freedom need to be considered.

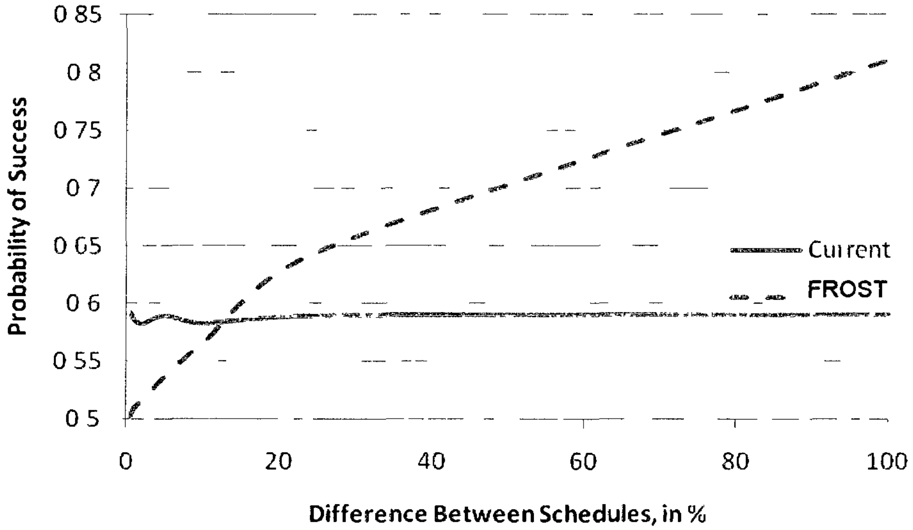
1. Number of parts in the system (P)
2. Years remaining in the system's lifetime (T)
3. How much failure data is available (more data = more accurate results)
4. The difference between schedules being compared (D)
5. The probability of successfully choosing the correct schedule (S)
6. FROST Method vs. old method

To consider these dimensions, 500 separate simulations were run for each combination of the following variables.

- P = 5, 25, 100 parts
- T = 2, 4, 8, 16, 32 years
- Failure data based on 1, 2, 4, and 8 failures per part, + based on vendor estimates
- D based around d=0.5, 1, 2, 5, 10, 25, and 100%, with the value of D being selected for each part P as a random variable uniformly distributed between 0 and 2d.
- Method = current, new

Each simulation for each of these 1050 combinations was tested with 1000 runs for the new method and 1 run for the old method, for each part, for a total of 22.8 million runs.

With all the scenarios averaged, the results between the two methods are as shown in Graph 8.



Graph 8 - Probabılıty of Determınıng Better Schedule, All Scenarıos

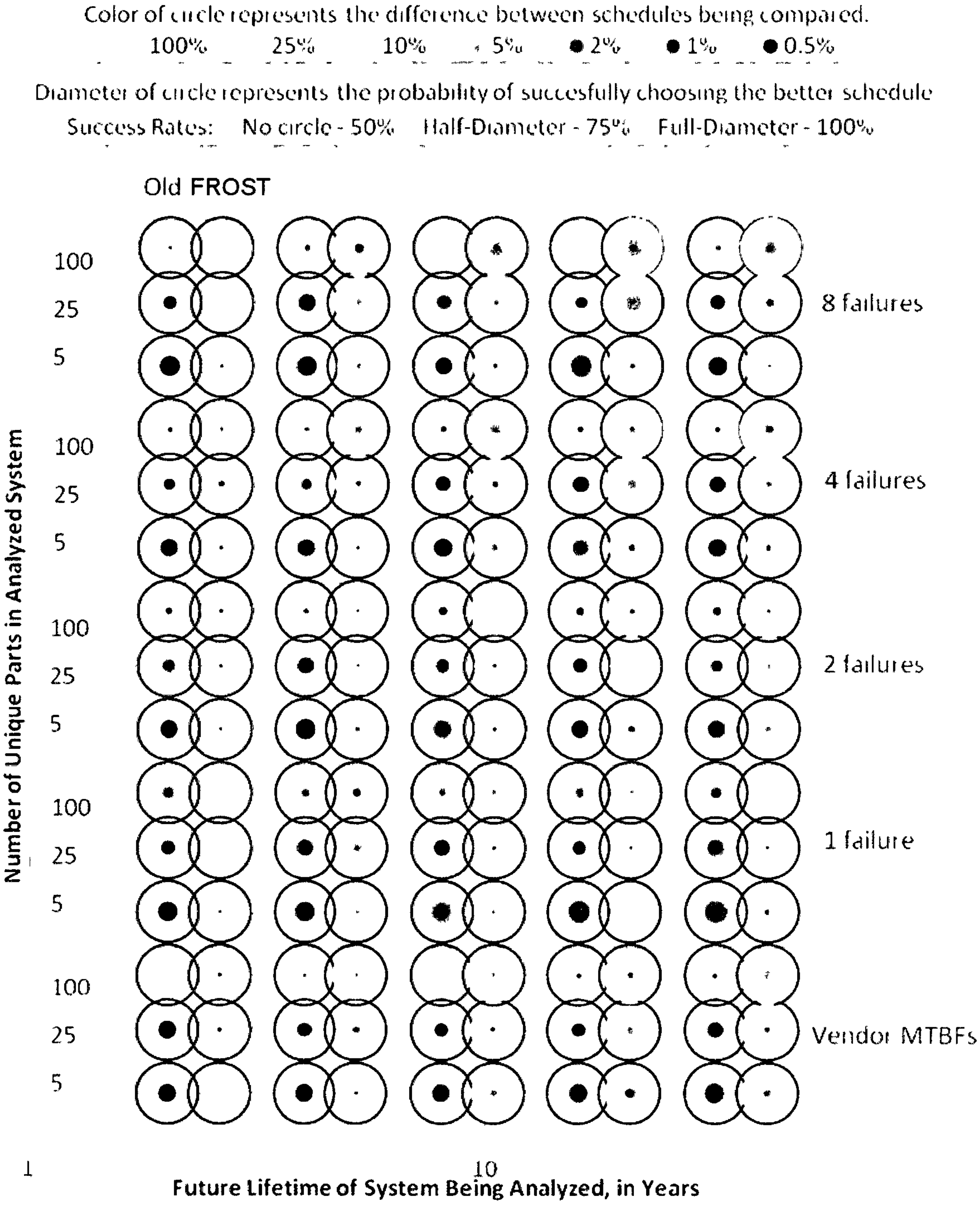
As would be expected, the proposed method converges at a 50% success rate at D = 0. In other words, if there is no apparent difference between the schedules, it is equally likely to pick either one. As the difference between schedules becomes more pronounced, it becomes more and more likely that the method will be able to successfully pick out the better schedule.

The current method, on the other hand, has a success rate which is mostly independent of the difference between schedules. It can either pick out the better schedule or it cannot. Through the various scenarios analyzed, it averaged a 58.7% success rate.

Some generalities can be observed from this graph. On average, for a small difference between schedules, the current method is actually better at picking out the preferred schedule. However, the current method is not able to provide a very high level of confidence for its choices.

In general, the utility of a method which can pick out a better schedule occurs for small differences between schedules. After all, if one schedule actually requires 100% more inventory to cover it, there is likely no need for a complicated method to tell managers that; it should be fairly obvious. Since on average the current method appears better at finding small differences between schedules, is the current method actually better? Not really. This is a result to be expected. The distributions and Monte Carlo process used in the FROST Method rely on randomness which effectively creates a small amount of noise. When the difference between schedules is small, in some scenarios this difference gets buried in this noise and the FROST Method has trouble picking it out, making it less accurate. However, this process also allows for a much better choice of solutions, which will be proven later. This better choice of solutions more than makes up for the inaccuracy. In effect, the current method has a uniform (~59%) chance of choosing the better schedule but results in inefficient solutions being used to sustain systems. The FROST Method, on the other hand, has varying accuracy depending on the scenario, but universally results in more efficient solutions. For example, it might choose a schedule which is 1% worse than optimum, but it will make up for this by reducing the cost to support that schedule by 40%.

Graph 8 showed the average result over a variety of scenarios, but as previously mentioned, there are 6 relevant factors which were tested in 1050 combinations. Since 1050 graphs would be a few too many to present, a 6-dimensional chart is provided to show the results. See Graph 9.



Graph 9 - Comparison Between New Method and Old Method for Successfully Choosing A Better Schedule

To use Graph 9, find the closest appropriate bubble for your system based on the number of parts, the remaining lifetime of the system, and the amount of failure data available using the left, bottom, and right axes, respectively. For failure data, the bottom-most section is to be used when the only failure data available is based on vendor claims, which are often wrong. The remaining sections use a Chi-Square distribution, as previously discussed in the distribution section, to model the success rate if all MTBFs were based on the corresponding number of failures.

Each bubble is composed of several circles of different sizes and color. A darker circle means the corresponding success rate can be achieved with a smaller difference between schedules being compared. A larger circle means a higher success rate, so a large dark circle means the method can successfully determine a better schedule with only a very small difference. A smaller or lighter circle means the method is less successful.

Bubbles come in pairs, with the left bubble representing the success of the current method and the right bubble representing the success of the FROST Method.

Several interesting observations are apparent from this graph. First, a quick visual scan shows that the Vendor MTBF results clearly fall between the results based on 2 and 4 failures. In other words, if 0, 1, or 2 failures of a particular part have been witnessed, a system is better off sticking with the estimated MTBF provided by the vendor instead of using real data. On the other hand, if 4 or more failures have been witness in the environment for a part, that data is of higher quality than that provided by the vendor. How it compares against 3 failures would be a matter for more study, but it appears it would be roughly a toss-up.

Second, the FROST Method's success increases as there are more parts in a system, but the current method's success decreases. As a result, in general the current method appears better for a system based on only a few parts, and the new method appears better for a system based on many parts. While seemingly counter-intuitive at first, this actually makes sense. The uncertainty caused by randomness actually increases with more parts for the current method but decreases for the FROST Method.

For the new method, the inclusion of the distributed inputs module introduces a huge amount of randomness, and that randomness is brought into check by using a large sample size. Increasing the number of parts essentially increases that sample size, further reducing the impact of that randomness and improving accuracy.

For the current method, this is not the case. Increasing the number of parts actually increases randomness. Consider the simplest case of a single part in the system. For a single part, it would be incredibly obvious using the current method which was the better schedule. If that part is used less on one schedule, that schedule will clearly be better no matter what and one could predict the better choice with 100% certainty. Add a second part, and things become less clear. That same schedule may be better for the first part, but worse for the second part. In this case, all of the randomness starts to be significant. Which part should be prioritized depends on how much it fails, which depends on the MTBF, which has randomness associated with it. It is no longer obvious which schedule is better, so by increasing the number of parts, the overall accuracy decreases. This explains why the old method performs better with fewer parts, and the FROST Method with more parts.

During the validation section, it was determined in equation 2.9 that the effectiveness of the FROST Method can be determined by calculating its reduction in cost.

$$Reduction\ in\ Cost = 1 - \frac{C_p \cdot S_p + C_p \cdot (1 + zD) \cdot (1 - S_p)}{C_c \cdot S_c + C_c \cdot (1 + zD) \cdot (1 - S_c)}$$

Simulations have provided values for the success rates of the two methods, S_p and S_c, under a wide variety of situations. These results are presented in the Results chapter. All that remains is to find values for the costs, C_p and C_c. And, in fact, this equation can be rearranged so that there is no need for actual individual costs but only a need for the ratio of the costs, C_p/C_c.

$$Reduction\ in\ Cost = 1 - \left(\frac{C_p}{C_c}\right) \frac{S_p + (1 + zD) \cdot (1 - S_p)}{S_c + (1 + zD) \cdot (1 - S_c)} \qquad (5.33)$$

It is now just a matter of finding out how the FROST Method compares cost-wise with the current method under a variety of scenarios, and the equation will be complete and able to be analyzed to show how much of an improvement it provides.

CHAPTER 6

COST MODELING EXAMPLE

In this section the Prediction of Costs and Sustainment Effort modules are discussed. The discussion of accuracy to this point has almost purely been based around inventory. In reality, the choice of the schedule should be based on cost. To truly choose the best schedule, the cost for supporting each part in the system must be determined.

The method up to this point has been generic. The distribution method, inventory prediction, and Monte Carlo process can all be applied to any electronic system. The conversion from the outputs of the Monte Carlo process to a cost, however, is somewhat system-specific. It needs to be tailored to the particular solutions, procedures, and costs associated with each system.

For example, for one system, substituting one part for another might be as simple as buying the new part and plugging it in. For another system, that same substitution might require hundreds of thousands of dollars of testing and documentation. This must be considered as the first system would surely make use of substitution much more often than the second system.

Additionally, the very solutions that are available might be different for different systems. For example, one solution used by the Navy is to make a deal with a vendor where the Navy will spend money to stockpile piece-parts and the vendor will maintain production abilities, and even after a part stops being produced for the general public, the Navy can exercise a contract where the vendor will use the Navy's stockpiled piece-parts to produce new copies of an otherwise obsolete part. A solution such as that would be outside the reach of many organizations and systems, but available for others.

Thus, these modules will be somewhat application specific and as a result every detail of the process cannot be provided in a manner that fits all possibilities. As a guideline, however, the method to create these modules will be provided. It should be evident how to modify the method for applications that use a different combination of solution types.

There are two situations under which solutions can be enacted: fully and situationally. Consider this example: 1000 runs are simulated for a part which is about to stop being produced. After analyzing those runs it is demonstrated that the best move is to purchase 10 parts before it stops being produced. In some runs more than 10 parts are needed and in others less, but the best action is to purchase 10.

Even though not all runs actually required 10 parts, since the decision needs to be made immediately with the best-available current data, those 10 parts will be purchased regardless of which future actually comes to pass. In other words, that purchase of 10 parts is executed fully, regardless of situation. If it turns out in the future that less than 10 parts were required, that does not change the fact that 10 have already been purchased.

Assume, however, that 10% of the runs simulated showed a requirement for more than 10 parts. In those 10% of futures, and only in those 10%, will a second, additional solution need to be enacted such as substituting in a different part. This is a situational solution, as it will only be enacted in certain future situations. Hence the two solution types, fully executed and situational.

These are separated because of their impact on accuracy. Consider if the initial purchase would cost \$10k and the second solution of a substitution would, if required, cost an additional \$50k to execute. Since simulations show a 10% chance of the second solution being required, the expected total cost would be

$$\text{Expected Cost} = \$10\text{k} + 10\% \times \$50\text{k} = \$15\text{k} \qquad (6.1)$$

Now assume that later on once better data becomes available and predictions become more certain, it becomes apparent that the chance of a substitution is 20%, not 10%, and the cost of that substitution has doubled to \$100k. At this point the \$10k purchase has already been made, so that cost does not change. The new expected total cost would be

$$\text{Expected Cost} = \$10\text{k} + 20\% \times \$100\text{k} = \$30\text{k} \qquad (6.2)$$

The important thing to note is that these two equations both have a term for fully executed solutions (the purchase) and situationally executed solutions (the substitution). The example inaccuracy in the cost modeling module made the situational term (the 10% x \$50k) change but did not have any effect on the fully executed term (the \$10k) because that cost has already been realized. This is a key point. Even if the costs for fully executed solutions turn out to be based on inaccurate methods or information, the costs themselves will still be accurate. This is because the information provided to the model, accurate or not, will be used to determine a solution. By the time it can be determined that the information was inaccurate, the solution will have already been executed and the money spent. This means that while inaccurate information might reduce the efficiency of the solution, it will not cause the estimated cost to be inaccurate; it will have accurately estimated an inefficient cost. In the example above, in hindsight it might turn out that based on better information 15 parts should have been purchased instead of 10, but it will not matter because 10 is what was purchased, and that cost will be accurately realized one way or the other. Therefore, regardless of the quality of predictions, fully executed costs will be accurate and only situational solutions will contribute to inaccuracy.

This is significant because the bulk of costs should fall into the category of fully executed solutions. More often than not, the most cost efficient solution will be to attempt to solve a situation up front, leaving a relatively small chance of needing to take further action later. For example, a set of simulations with starting inventory of zero, lifetime of 8 years, and less than 100 typical failures was run. An analysis of the results showed that fully executed solutions accounted for 84.8% of the total cost, with a variance of 13.4%. This means that the costs which are actually susceptible to inaccuracy are only a small fraction (15.2%) of the total cost and therefore the total cost will be very tolerant to inaccurate inputs without a large effect on overall accuracy. In summary, inaccurate inputs will mostly affect the FROST Method by making it less efficient, not less accurate.

With that concept in mind, the next step is to detail the process which will provide these solution choices and cost estimates. There are four necessary inputs for this process.

6.1 Required Inputs

1) Identification of Solutions to Consider
2) Identification of Fallback Solutions
3) Cost Factors
4) Simulations of Future Inventory

Identification of Solutions to Consider is a list of potential ways that you can prevent a part from running out of inventory. For example: purchasing new parts, repairing failed parts, substituting a different part, redesigning the part out of the system, etc. This can vary from system to system, depending which solutions are available.

Identification of Fallback Solutions is identification of which solutions, during the lifetime of the system, do not have a time at which they stop being viable. For example, purchasing parts is generally not a fallback option because at some point the vendor will stop producing and selling the part and it will no longer be possible to "fall back" on that solution. On the other hand, redesigning a system can be done at any time in the future, making it a viable fallback solution.

Cost Factors are basic factors which are sufficient to estimate the cost of the identified possible solutions. For example, consider the following factors:

a) $800 - Labor cost to place an order
b) $1400 – Cost per part charged by vendor
c) $100,000 – Cost to test and verify a substitute part
d) $1200 – Cost per substitute part charged by vendor

From this, it could easily be calculated that to order 10 parts would cost $800 + 10 x $1400 = $14,800. It could also be seen that substituting a new part and ordering 10 would cost $800 + $100,000 + 10 x $1200 = $112,800.

Simulations of Future Inventory are predictions of future inventory levels at any point in time if no solutions are enacted. This is the output of the Prediction of Health module and the Monte Carlo process outlined earlier.

6.2 Steps

The following steps, when followed, determine the best solution or combination of solutions for a part. They also find the anticipated cost of those solutions, fulfilling the requirements for the Cost Prediction and Sustainment Effort modules.

1) Find the Minimal Cost to implement each type of solution to be considered
2) Determine the Sufficiency of each type of solution
3) Find the Estimated Inventory Shortfall
4) Find the Estimated Solution Cost for each solution type in order to overcome the Estimated Inventory Shortfall
5) Predict which Solutions will be used based on Estimated Solution Costs and Sufficiency
6) Calculate Fallback Chance and Optimal Actions to Take

1. Find the Minimal Cost to implement each type of solution to be considered

This step involves using cost factors to calculate the cheapest possible implementation for each possible solution. For example, the cheapest implementation of a purchase would be to complete an order for a single part.

2. Determine the Sufficiency of each type of solution

Sufficiency for a solution, in this context, means that if the solution in question were enacted it would not make sense to enact any additional solutions. This sufficiency is to be determined for each solution, and additionally for each combination of solutions. In this method, sufficiency is approximated using the following calculation.

FallbackChance = The fraction of simulations where inventory drops below zero if the solutions being evaluated are enacted to their fullest possible measure.

MinCost = The Minimal Cost to implement an additional solution. In other words, it is the cheapest implementation of the cheapest solution not being evaluated for sufficiency.

FallbackCost = The Minimal Cost to implement an additional fallback solution. In other words, it is the cheapest implementation of the cheapest fallback solution not being evaluated for sufficiency. This does not have to be a different solution than the one found in MinCost.

The solution being evaluated is assumed to be sufficient if and only if the following is true

$$\text{MinCost} > \text{FallbackCost} \times \text{FallbackChance} \qquad (6.3)$$

This is an extremely good approximation of sufficiency. Imagine you are testing the sufficiency of harvesting. Simulations show that in 90% of runs, you will not need another solution if you implement harvesting. This makes the FallbackChance 10%. The question then is whether it is worth adding another solution on top of this to further reduce that fallback chance. Imagine that you only have one fallback solution available and the cheapest implementation of it costs $1M. The expected value of this fallback solution would be 10% x $1M = $100k. If there is no way to improve this situation that is cheaper than $100k, then it does not make sense to take further action at this time and harvesting is sufficient.

The only issue with this simplification is that, while it is true that a solution *more* expensive than $100k should not be implemented, the corollary is not always true that a solution less expensive than $100k should be implemented. There is no guarantee that enacting another solution will reduce the fallback chance a significant amount. It is possible that throwing another $50k at the problem would only reduce the fallback chance 1%. In that case, $50k was just spent to reduce the expected cost by $10k. In practice, this is almost never the case and the equation correctly determines sufficiency. In the rare cases that it does incorrectly determine sufficiency, however, a future step will solve this issue.

3. Find the Estimated Inventory Shortfall

This step involves finding the minimum balance of inventory seen at any time in each simulation run, and then averaging. The formula is as follows

$$Shortfall = -\sum(MinInventoryBalance \cdot ProbabilityofSimulation) \quad (6.4)$$

4. Find the Estimated Solution Cost for each solution type in order to overcome the Estimated Inventory Shortfall

Use Cost Factors to determine the cost, for each solution type, which would improve inventory enough to overcome the Estimated Inventory Shortfall. For example, if there was a shortfall of 8.5 parts, then for the solution type of purchase, this would be the cost to place an order for 9 parts.

5. Predict which Solutions will be used based on Estimated Solution Costs and Sufficiency

Using the Estimated Solution Cost for each solution and the Sufficiency for each solution, evaluate whether a solution will be used. This step should determine whether a solution will definitely be used (100% of the time), will not be used (0% of the time), or will be used as a fallback solution only if necessary (between 0 and 100% of the time). The concept employed is that the best choice is the cheapest combination of solutions which results in sufficiency.

The simplest way to accomplish this is, for each solution or combination of solutions which is sufficient, to add up the total Estimated Solution Costs for those solutions. Whichever combination results in the cheapest total cost is the correct choice. Therefore, the solutions it finds will all be used 100% of the time. Any other solutions are noted as being used 0% of the time.

If the combination of solutions found does not already involve a fallback solution, identify the cheapest available fallback solution as being used as a fallback solution.

Provided below is a possible example of a logic chain that could be used to accomplish this step. In this example, the system in question uses Substitutions, Harvests (re-use of parts), Aftermarket Repair Depots, and Redesigns as solutions, with Substitutions and Redesigns both qualifying as fallback solutions.

- Substitution is used 100% of the time if it is cheaper than all other available solutions
- Substitution is used 100% of the time if it is cheaper than all other available solutions except harvesting, but harvesting is not sufficient
- Substitution is used 100% of the time if it is cheaper than all other available solutions except aftermarket repair depot, but aftermarket repair depot is not sufficient
- Substitution is used 100% of the time if it is cheaper than all other available solutions except aftermarket repair depot and harvesting, it is cheaper than the combination of both aftermarket repair depot and harvesting, and neither aftermarket repair depot nor harvesting are sufficient by themselves

- Substitution is used 100% of the time if it is cheaper than all other available solutions except aftermarket repair depot and harvesting, but aftermarket repair depot combined with harvesting is not sufficient
- Substitution is used as a fallback option if none of the above situations are true and Substitution is cheaper than Redesign
- Substitution is used 0% of the the time otherwise

This would correctly determine whether or not Substitution would be used as a solution for a part. A similar logic chain would be needed for each solution type. Code which fully determines use is available in the appendix for the following solutions: buy, harvest, aftermarket repair depot, redesign, and substitution.

6. Calculate Fallback Chance and Optimal Actions to Take

Fallback Chance is the probability that the fallback solution will be used. To find this probability expected costs must be minimized.

$$Expected\ Cost = \sum CostOfSolution \cdot ProbabilityOfSolutionBeingUsed \tag{6.6}$$

Since at this step the probability of every solution except the Fallback Solution is already known, this reduces to

$$Expected\ Cost = FallbackSolutionCost \cdot FallBackChance + \sum CostsOfFullyUsedSolutions \tag{6.7}$$

In this equation, most solutions will have a non-variable cost, such as when performing a redesign. Some solutions will have variable costs, such as performing a purchase. In that case, the cost depends on the number of parts purchased, X_1. There could also be X_2, X_3, etc, if there are multiple solutions being used with variable costs. So, the Costs of Fully Used Solutions is a function of these variables, reducing this equation to

$$Expected\ Cost = \\ FallbackSolutionCost \cdot FallBackChance + \ CostsOfFullyUsedSolutions(X_1, ..) \tag{6.8}$$

The Fallback Solution Cost is already known, and the Fallback Chance is found by determining the fraction of simulations, once the chosen solutions are enacted, which result in inventory falling below zero. This is dependent on the exact same variables as the costs. For example, the more parts which are purchased, the smaller the chance inventory will run out and a Fallback solution will be used. Thus, the entire equation is only dependent on a couple of variables. At this point, standard iteration techniques are used to determine which combination of values for these variables results in the smallest Expected Cost. This results in knowledge of what solutions should be used, any variables involved such as how many parts to purchase, what the chance is of these solutions failing long-term, and what solution should be used it that failure occurs.

With that, the Sustainment Effort and Prediction of Costs modules are created. This completes the definition of the proposed method. All that remains is to show its effectiveness compared to current methods.

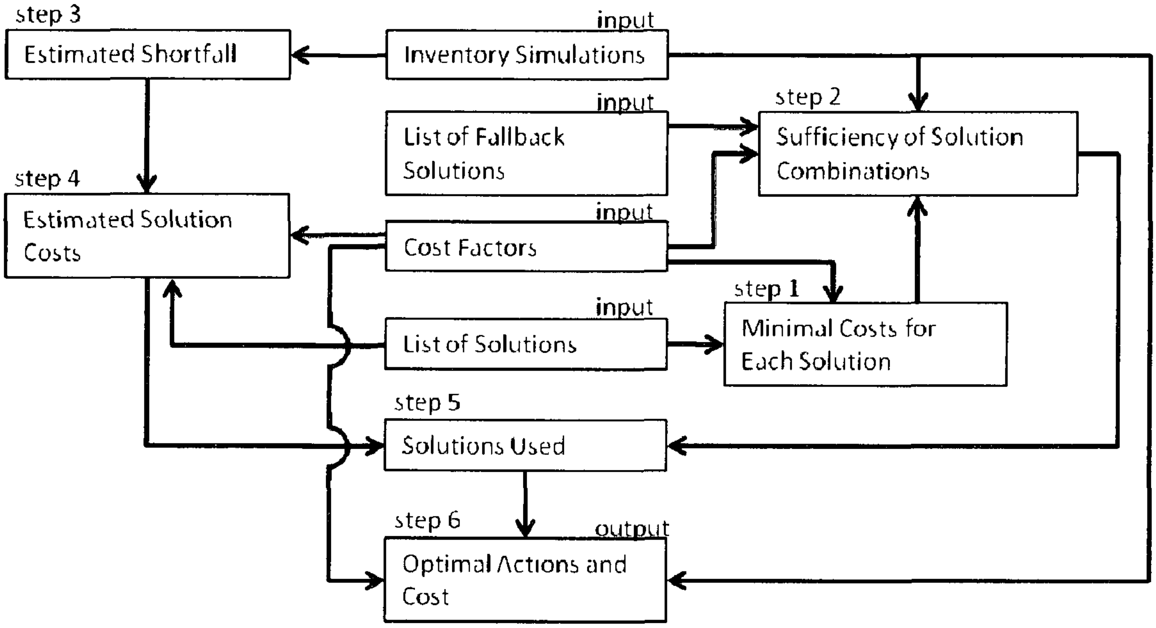


Figure 16 - Flowchart for Cost/Solution Modules

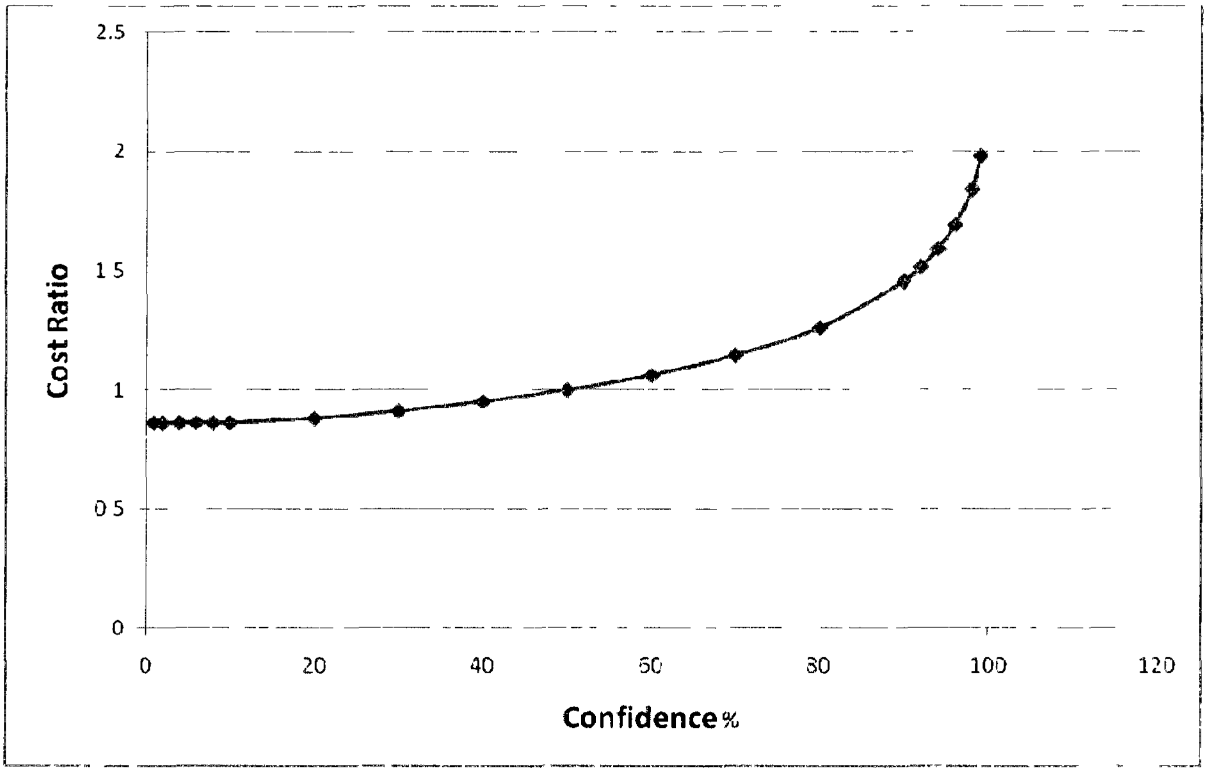
6.3 Comparison Between Methods

With this process determined, several sets of simulations were run to compare the performance of four different methods in different scenarios. The first method uses the newly developed process outlined above. The other three methods are all based around the single-run, non Monte Carlo process currently being used as standard in industry and research. In these methods, instead of predicting a variety of futures based on the fact that inputs such as failure rates and end of production dates are probably inaccurate, a single future is predicted based on the assumption that the inputs are accurate.

Given this single-prediction process, there are still a variety of ways to react to these predictions. I have chosen three ways in particular to use as comparisons.

The first way is the simplest. It is referred to as the "Buy @ 50%" method. Simply put, while a system still has the option, its managers buy the parts they expect to need. So, if their MTBF, usage rates, etc. combine to say they most likely going to need 100 parts, they simply buy 100 parts. It is referred to as the "Buy @ 50%" method because, at least as far as this method is capable of determining, there is a 50% chance this will be enough parts. Whenever this buy turns out not to have been sufficient, there is a future cost to perform a redesign or substitution.

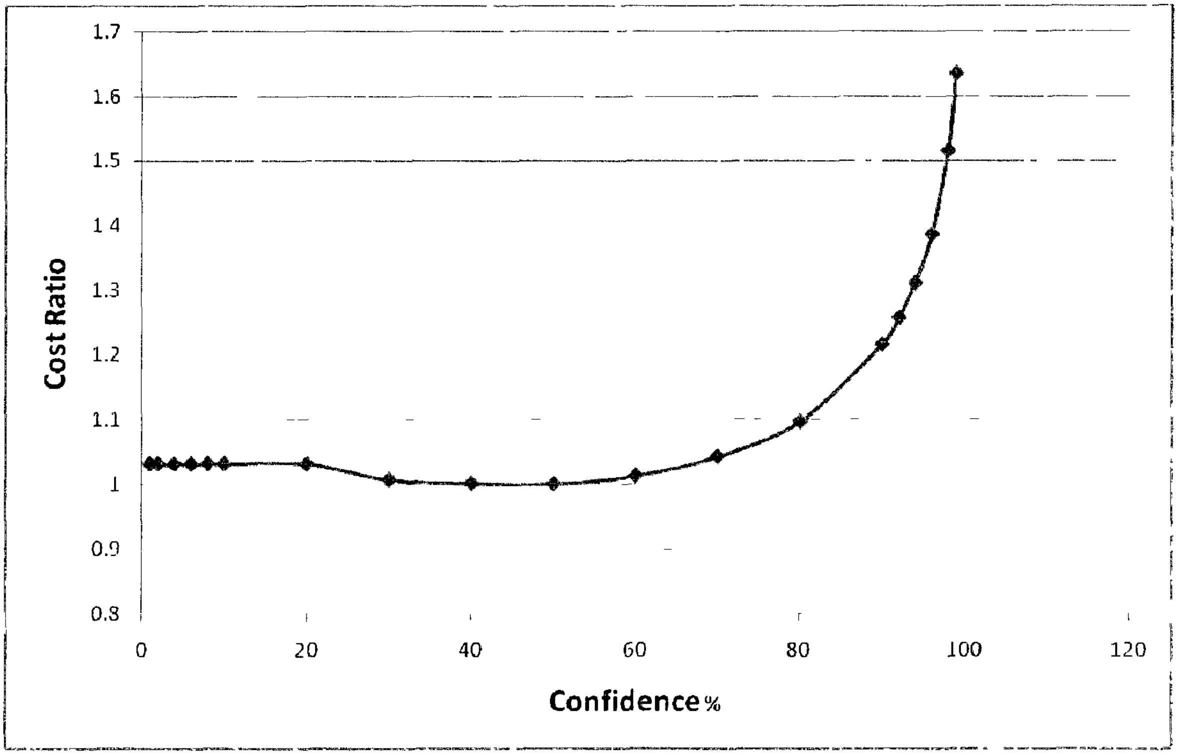
The next question is whether 50% is the right confidence at which to buy. There are well-documented equations to determine a likely distribution of failures without performing a full Monte Carlo analysis. As discussed previously, the inverse of the Gamma Distribution can be used to calculate the expected number of failures. This same equation can easily be set to calculate at probabilities other than 50%. I ran a set of simulations for a scenario where there is a system with lifetime of 10 years, there are typically less than 100 failures, and inventory is already possessed at 80% of the required levels, and attempted buys at different probabilities than 50% to determine the ideal. Results from this scenario are not necessarily reflective of results in other scenarios, but this should at least provide a baseline. The results are shown in Graph 10.



Graph 10 - Cost vs. Confidence for "Buy @ Confidence" Methods

The results on this graph were normalized so that the cost at 50% confidence equaled one. In this scenario, a lower confidence is apparently better. In fact, the 6 lowest confidences tried (1, 2, 4, 6, 8, and 10%) all resulted in a cost of 86.2% of buying at 50% confidence. While this will not be true for all parts and all scenarios, it seems that buying just enough to be alright in a small fraction of future scenarios, on average, provides better results than spending the extra money to get a higher confidence up front. Thus, the third method for comparison is "Buy @ 5%."

For the fourth method, the simulation was made to run a smarter solution method than to simply purchase the required number of parts. Instead, it assumed the best-case scenario where a team of people would attempt to find out the cheapest way to acquire enough inventory to meet projections. This includes any combination of harvesting, setting up repair depots, substituting, setting up aftermarket production, or purchasing additional inventory. This was run at a 50% confidence and then, as with the "Buy" methods, this same solution was tested at other confidences.



Graph 11 - Cost vs. Confidence for "Best @ Confidence" Methods

The results show that under this method, the 50% confidence level actually is the best level.

This leaves 4 methods to model for comparison:

- FROST Method – This is the method proposed in this dissertation. It uses distributed inputs and a Monte Carlo process to predict possible futures, then chooses a combination of solutions to minimize the immediate and future costs.
- Best @ 50% - This is the best possible result using the current method of relying on inaccurate data to predict a single future. This is actually a better, more in-depth process than is typically used, but since it represents the best possible outcome using current methods it will be used as the baseline. This is what the FROST Method will be compared against.
- Buy @ 50% - This is the simplest method where the expected inventory requirement is simply purchased, without analyzing other solutions until a problem arises. Some groups use better methods than this so it will not be used as the baseline, but this is probably the most common method used right now due to its simplicity.
- Buy @ 5% - This is the second simplest method, where buying more parts is still the standard solution. This represents the best typical outcome of a "Buy" method, although one that is rarely if ever actually used.

With the four methods determined, 20,000 simulations were run for each of these 4 methods for 32 different scenarios, giving 2.56 million data points. The variables tested were the lifetime of the system, the amount of inventory that is already on hand, and the typical number of failures seen in the system. A summary of the results is shown in

Table 6. From this table, known values for any system can be used to estimate the results of the FROST Method.

Life	Starting Inventory	Failures	New	50% Best	50% Buy	5% Buy
2	0	10	0.620	1	1.028	1.339
2	0	100	0.650	1	1.297	1.217
2	0	1000	0.606	1	1.509	1.197
2	0.5	10	0.437	1	1.010	1.262
2	0.5	100	0.649	1	1.198	1.089
2	0.5	1000	0.608	1	1.417	1.015
2	0.8	10	[illegible]	1	0.970	1.070
2	0.8	100	0.784	1	1.104	1.010
2	0.8	1000	[illegible]	1	1.281	[illegible]
4	0	10	0.632	1	1.119	1.415
4	0	100	0.664	1	1.419	1.294
4	0	1000	0.676	1	1.560	1.290
4	0.5	10	0.470	1	1.082	1.396
4	0.5	100	0.547	1	1.324	1.131
4	0.5	1000	0.525	1	1.491	1.049
4	0.8	10	0.351	1	1.011	1.159
4	0.8	100	0.709	1	1.203	1.024
4	0.8	1000	[illegible]	1	1.341	0.964
8	0	10	0.599	1	1.330	1.541
8	0	100	0.702	1	1.545	1.392
8	0	1000	0.731	1	1.602	1.367
8	0.5	10	0.482	1	1.283	1.601
8	0.5	100	0.482	1	1.462	1.203
8	0.5	1000	0.514	1	1.543	1.138
8	0.8	10	0.407	1	1.118	1.222
8	0.8	100	0.600	1	1.336	1.038
8	0.8	1000	0.637	1	1.400	0.975
16	0	10	0.633	1	1.454	1.537
16	0	100	0.763	1	1.621	1.464
16	0	1000	0.775	1	1.653	1.452
16	0.5	10	0.427	1	1.375	1.489
16	0.5	100	0.521	1	1.540	1.268
16	0.5	1000	0.571	1	1.600	1.237
16	0.8	10	0.498	1	1.249	1.398
16	0.8	100	0.511	1	1.388	1.046
16	0.8	1000	0.507	1	1.448	0.994

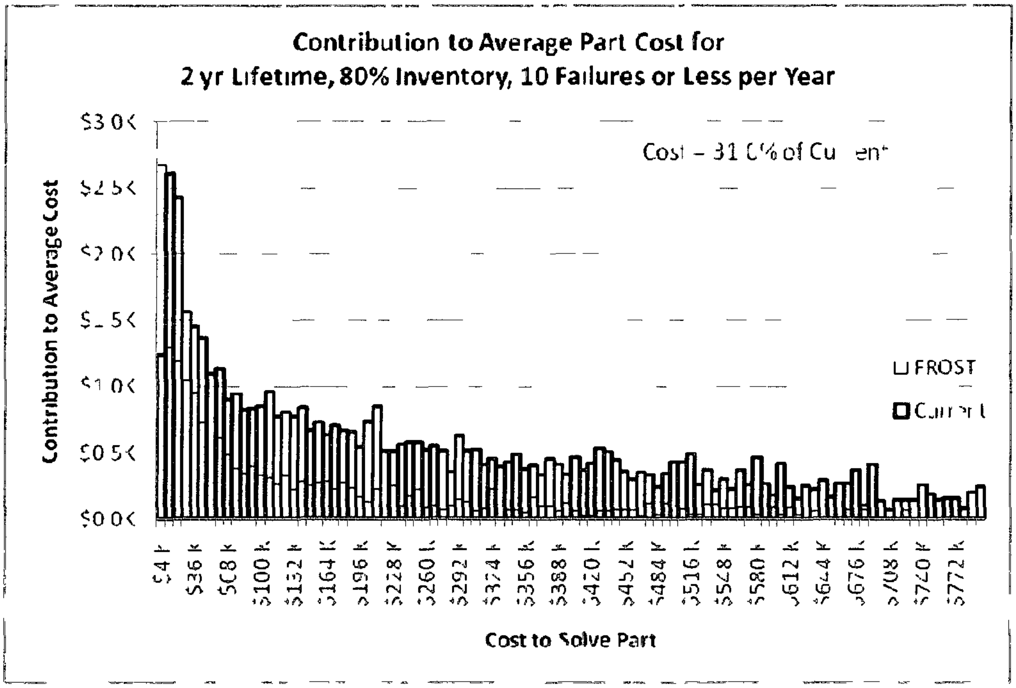
Table 6 - Cost Ratio Compared to "Best @ 50%" Method

In all scenarios tested, the FROST Method outperformed the baseline (Best @ 50%) method. The FROST Method ranged from 31.0 to 96.8% of the cost of the baseline. The scenario where the 96.8% cost occurred seems to be an exception to the standard performance. In this situation, the remaining lifetime is very short (2 years), there are a tremendous number of failures (up to 1000 per part per year) and the system already has most of the inventory it will ever need (80% of the requirement). This is not a likely scenario, but it is possible, and it turned out to be the only scenario that could be bested by a method that does not consider. The Buy @ 5% method in this situation outperformed it by 1%. This scenario would likely only be realistic for a massive installation nearing the end of its life. In the other 31 scenarios, the new method outperformed the baseline by only costing between 31.0 and 79.1% as much.

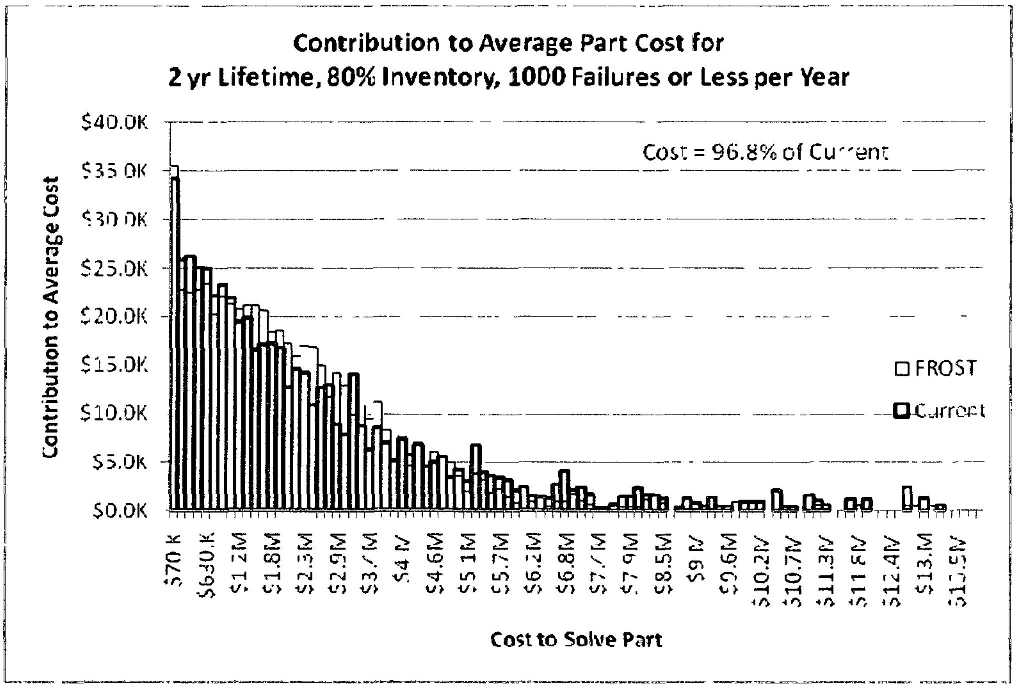
The Best @ 50% method almost always beats the Buy @ 50% method, with the one exception being very short lifetime, very high inventory, and very few failures. The Buy @ 5% method outperformed the Buy @ 50% method when there were 100 or 1000 failures typically, but performed worse when only 10 failures occurred.

The results show that the FROST Method would outperform all currently possible methods in all realistic scenarios. The following charts further show the results for each scenario. Three scenarios are included; the first shows the best case scenario, the second shows the worst case scenario, and the third shows a scenario which closely matches an actual system I am familiar with and is representative of average performance. The remaining charts are available in the appendix.

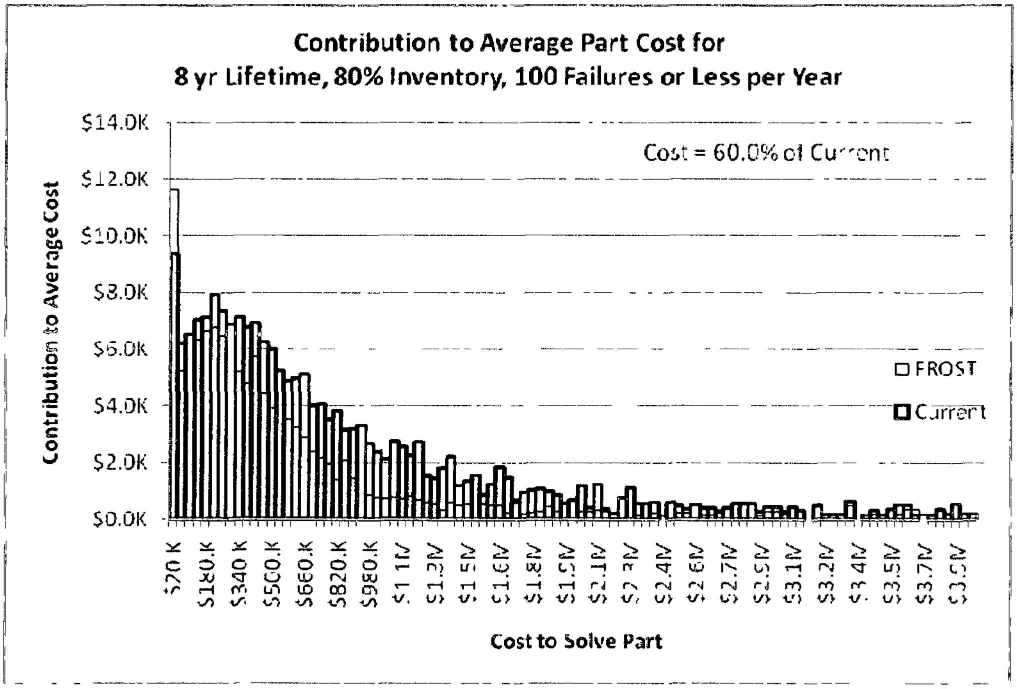
In these charts, a histogram showing the frequency of solutions at different costs was multiplied by those costs. This results in a graph where the area is proportional to the overall cost. In general, the FROST Method is similar on the left side of the graph, but shorter on the right side of the graph. This indicates that just as much money is being spent on inexpensive solutions, but less is being spent on expensive solutions.



Graph 12 - Contribution to Cost vs "Best@50%" Method, Best-Case Scenario

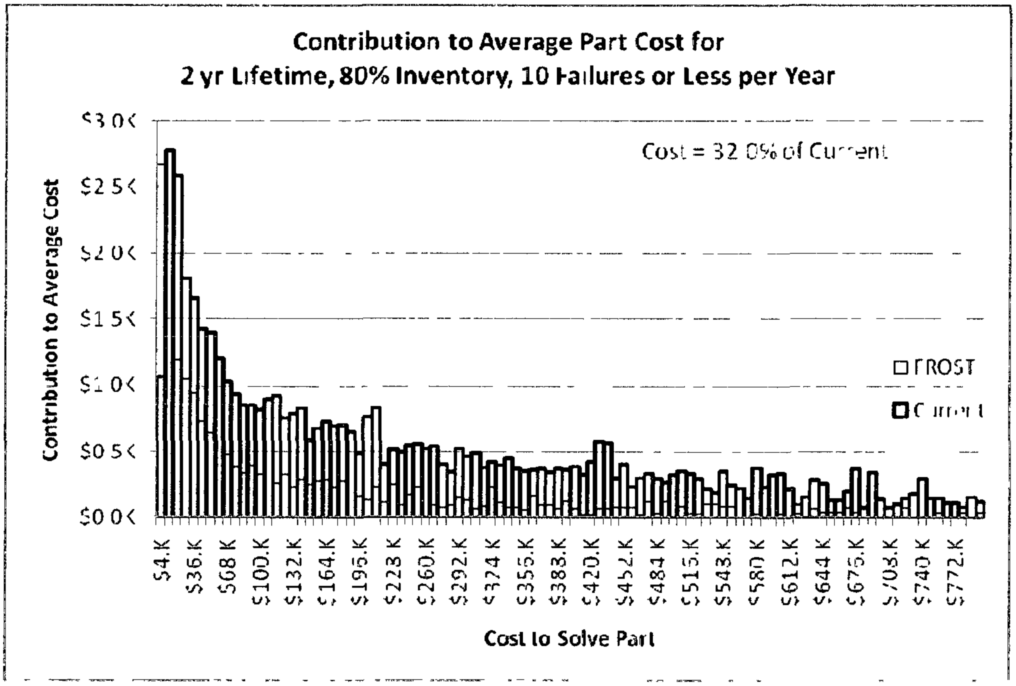


Graph 13 - Contribution to Cost vs. "Best@50%" Method, Worst-Case Scenario

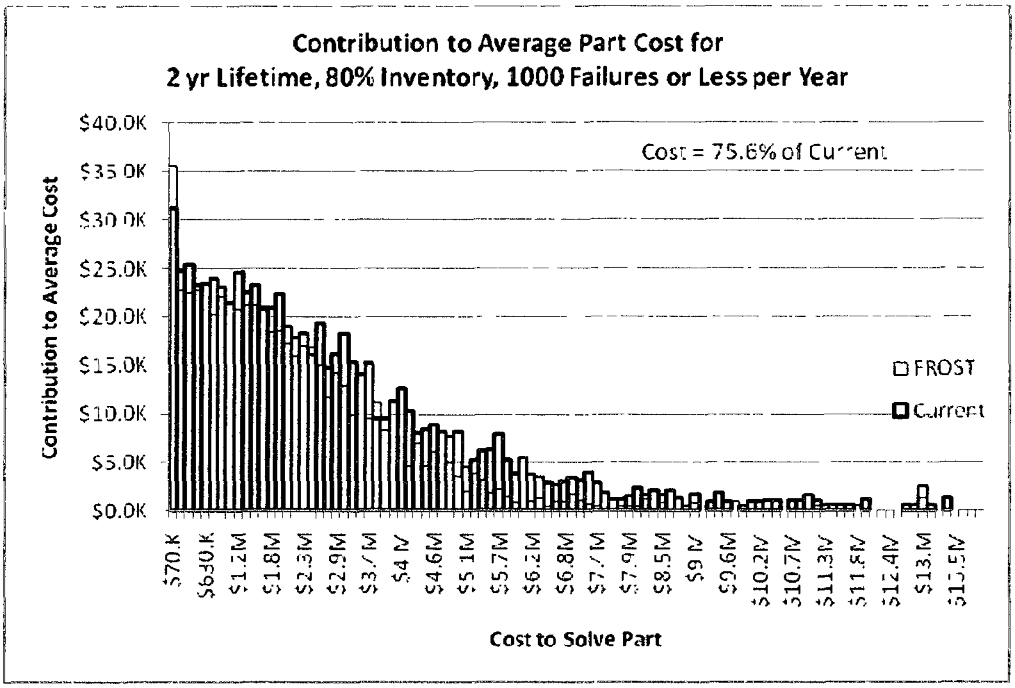


Graph 14 - Contribution to Cost vs. "Best@50%" Method, Typical Scenario

While the charts and numbers used throughout the rest of this paper are based on the standard baseline of the Best @ 50% method, Graph 15 and Graph 16 below show the best and worst scenario difference between the proposed method and the Buy @ 50% method which is also fairly common.



Graph 15 - Contribution to Cost vs. "Buy@50%" Method, Best-Case Scenario



Graph 16 - Contribution to Cost vs. "Buy@50%" Method, Worst-Case Scenario

CHAPTER 7

INDEPENDENCE OF SCHEDULE AND COST PREDICITONS

Having shown that for cost and solution prediction purposes the FROST Method is a significant improvement, the next question is whether the results of these simulations are applicable universally, or if these results are only a function of the way I chose to set up the simulations.

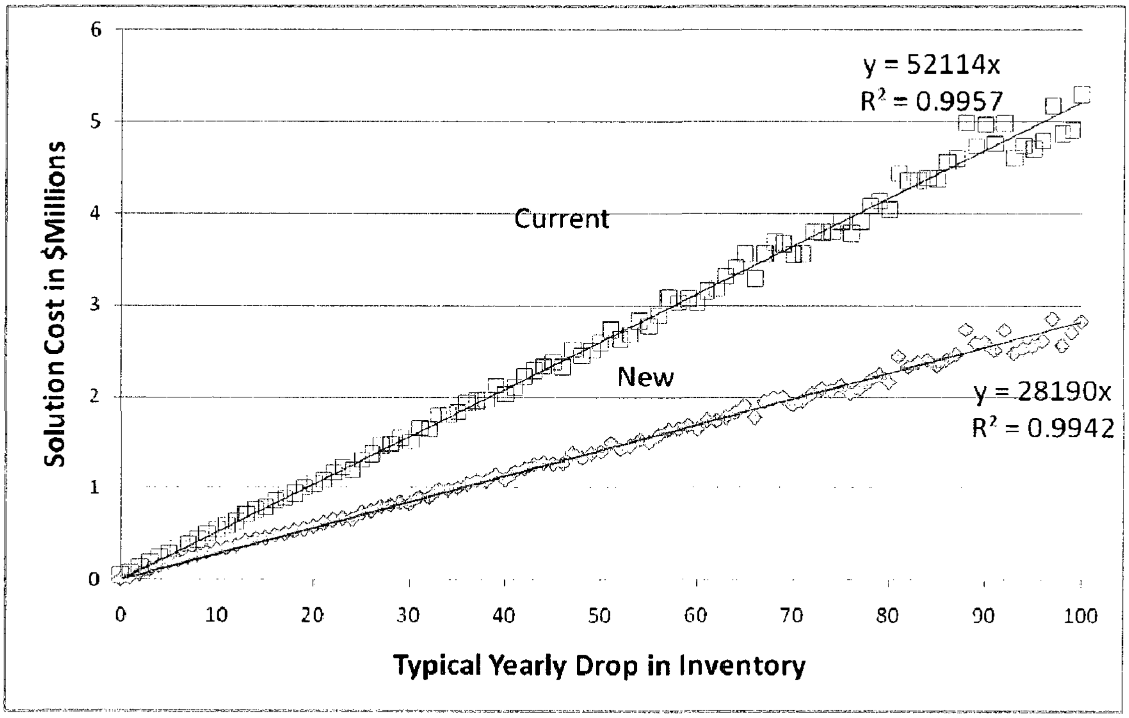
System lifetime, the number of failures, and starting inventory are all significant factors that impact the results. But do other factors matter? Here are the relevant variables and the assumptions made about their values in the simulation code. Insignificant variables, such as ones that would account for $100 of a $50k solution, were present in the code but not included here.

```
BuyYearsRemaining = 8 * Rnd() ^ 3

HarvestItemCost = 400 + 1200 * Rnd()

SubsYearsExtended = 1 + 4 * Rnd() ^ 2

SubsTestingCost = 1100000 * (Rnd() ^ 5) + 1000

SSBSetupCost = 25000 + 50000 * Rnd()

RedesignDesignCost = 10000 + 2200000 * Rnd() ^ 5

DepotSetupCost = 25000 + 25000 * Rnd()
```

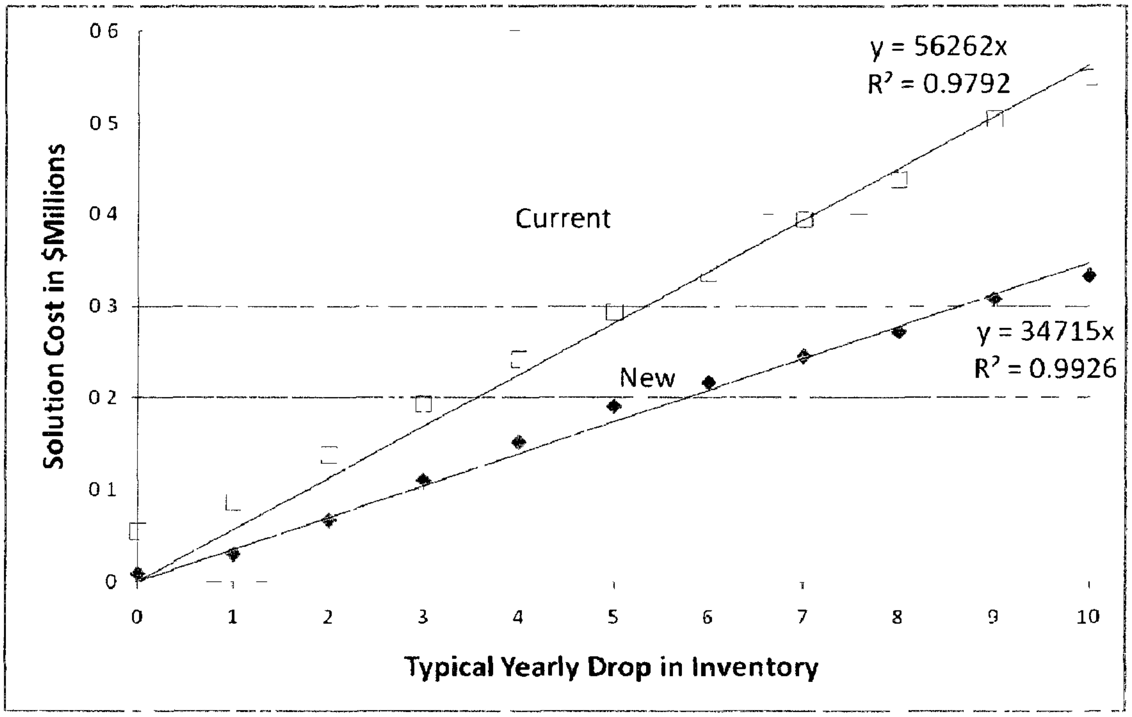
The important issue to note is that these are fairly rough choices for values. For example, the first item represents the number of years remaining to buy a part before it goes out of production. Were additional research and data-mining conducted, the result would surely show that these values in real life do not closely fit 8 times a uniform distribution from 0 to 1, cubed. Based on my own personal knowledge I feel that this is a reasonable approximation, but I acknowledge it probably would not have an extremely high R^2 value if it were compared against real data. So, had the extra effort been taken to more accurately choose values for these variables, would the results have been different? If it turns out that these choices for values have a large impact on cost, then the results cannot safely be assumed to apply universally. It would suggest that the results are not accurate since my choice of distribution is impacting the results. Further, since these distributions could possibly be system-dependent, it could mean that the results are also system dependent, and not universally applicable. However, on the other hand if it turns out that these variables do not have a significant impact on cost, then it indicates the results are accurate and applicable regardless of system. The only variables that were isolated in the results were lifetime, the number of failures, and starting inventory, and other variables were treated as insignificant.

To show this to be a correct path, a sensitivity analysis was run for all of these variables as well as for the typical yearly drop in inventory. Millions of simulations were performed over a wide variety of scenarios, and the resulting costs are plotted against the different inputs. Recall that earlier Condition 1 stated that for the final comparison method to be valid, the cost has to be essentially proportional to the drop in inventory.

For the first sensitivity analysis, the drop in inventory was compared against cost. Since this depends on what range of data is looked at, this was done for both for 0-100 parts per year, and 0-10 parts per year.

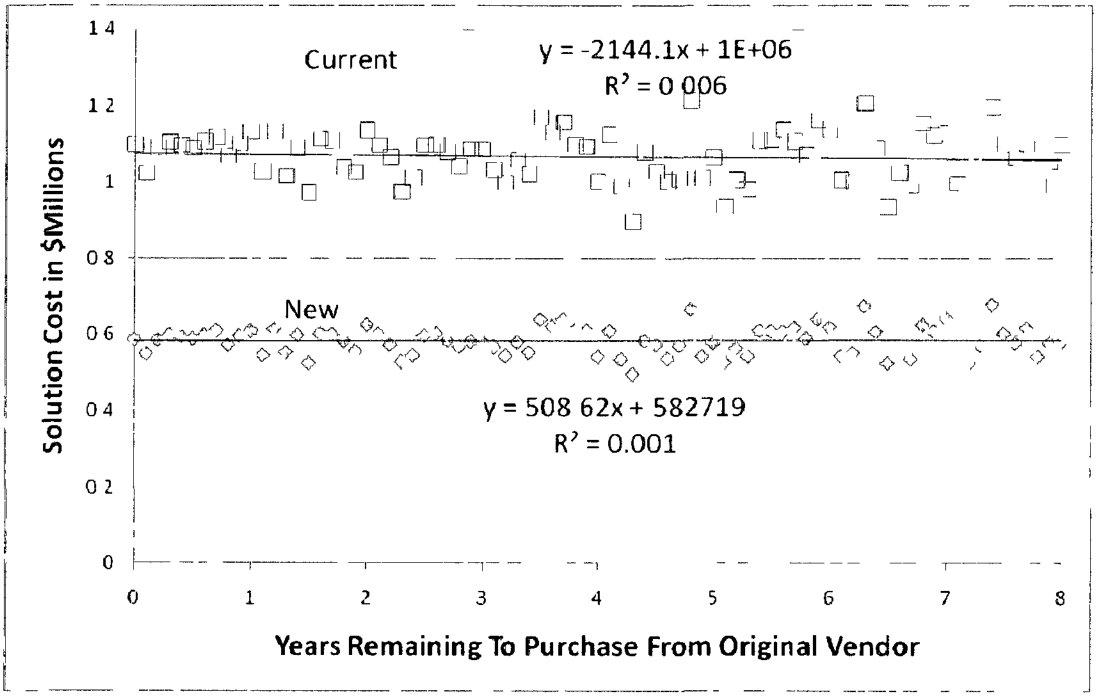


Graph 17 - Sensitivity Analysis for Cost vs. Yearly Drop in Inventory, 0-100

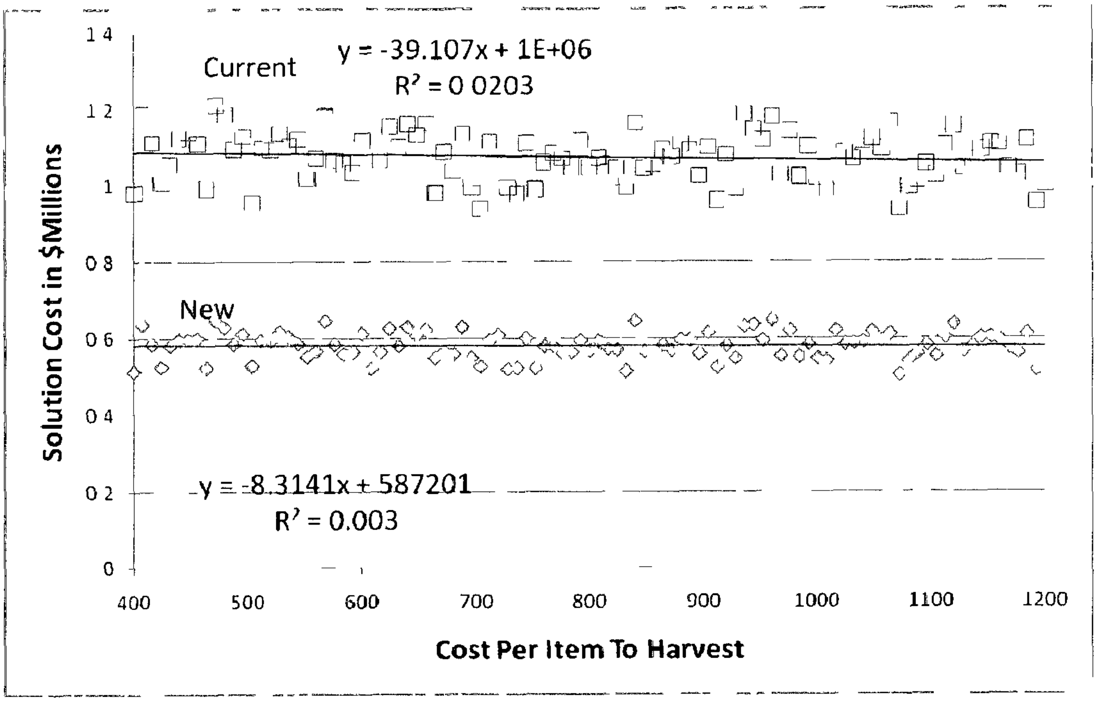


Graph 18 - Sensitivity Analysis for Cost vs. Yearly Drop in Inventory, 0-10

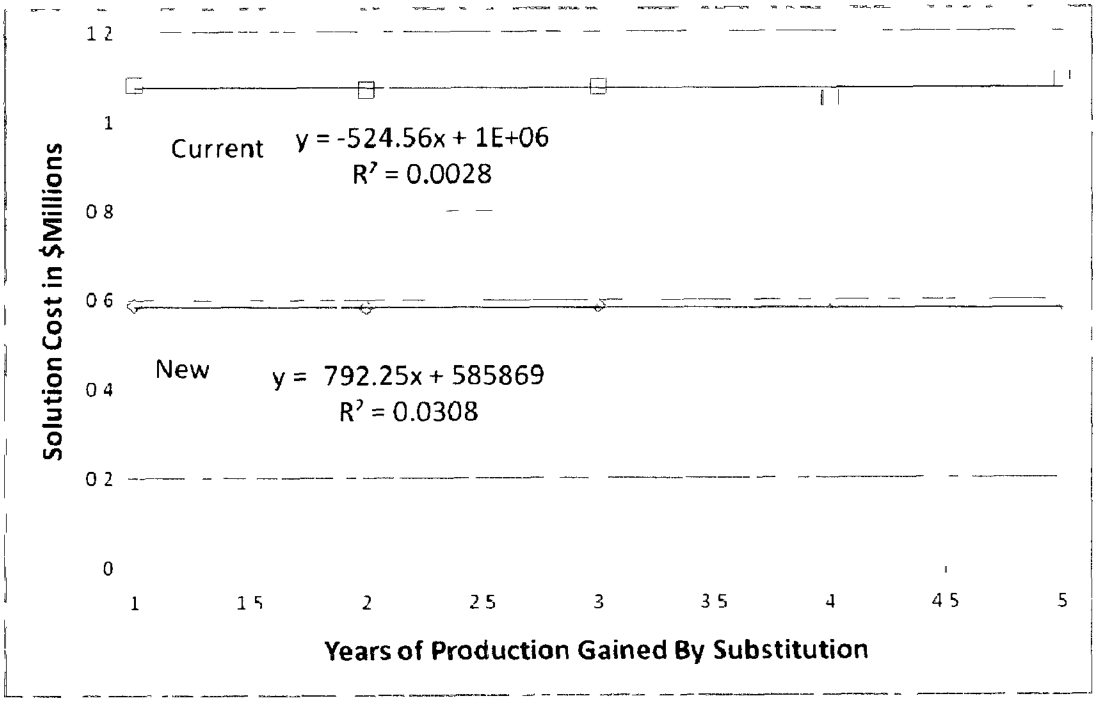
In both ranges, the relationship is quite linear. Both were fitted to lines with an intercept of zero. For the smaller range, 0-10 failures per year, this linear relationship had an R^2 value of 0.9926. As the range grew, so did the accuracy, giving the 0-100 range an R^2 value of 0.9942. This shows that the assumption in the scheduling section that cost could be modeled as proportional to drop in inventory is a good one, and condition 1 is met. Along with conditions 2 and 3, all requirements have now been met for the cost reduction equation to be validated. As long as the underlying assumptions are met the results presented in this dissertation can now be safely viewed to hold true. The only remaining concern is whether cost is dependent on any of the variables that were not isolated. The following set of charts shows that none of these variables has a correlation to be concerned about.



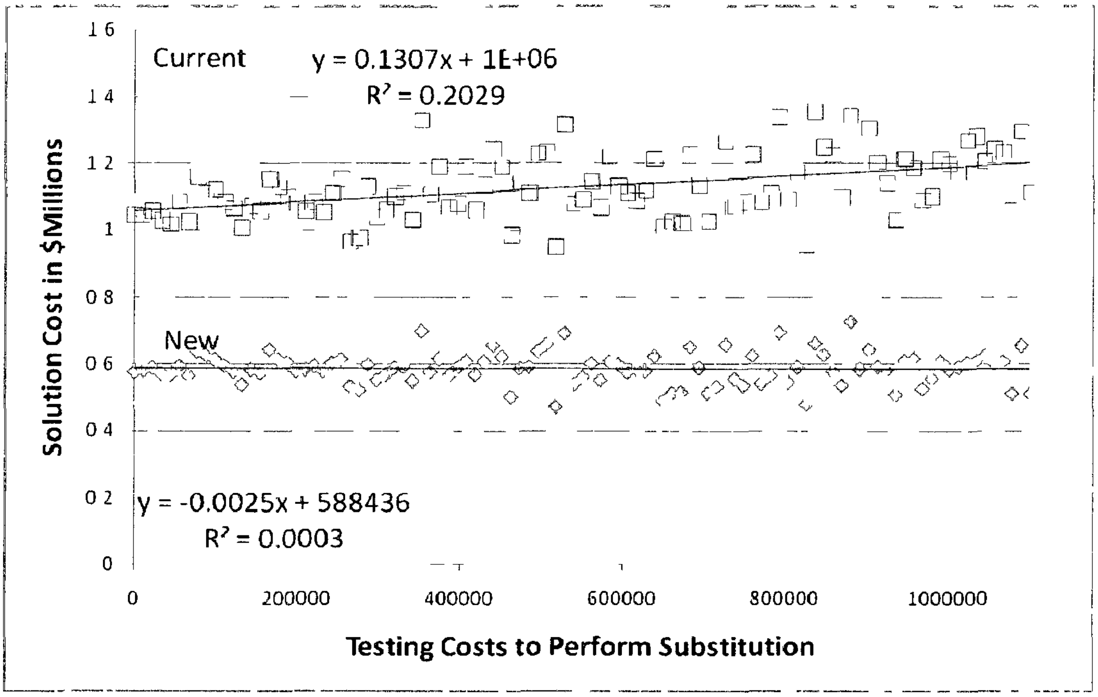
Graph 19 - Sensitivity Analysis for Cost vs. Remaining Production



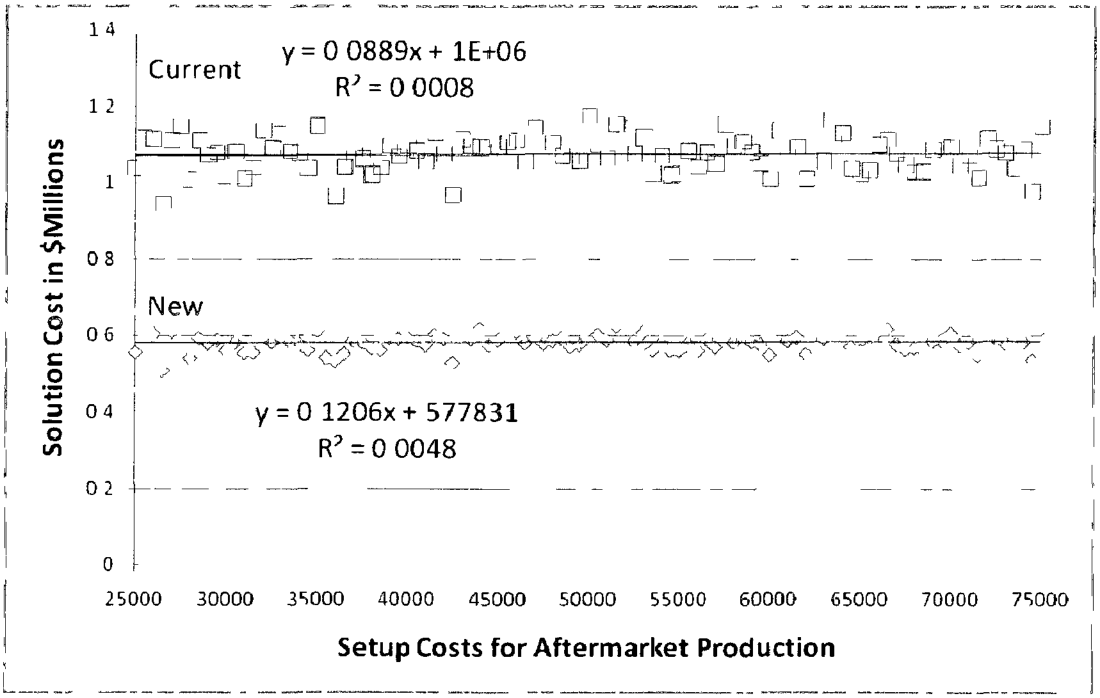
Graph 20 - Sensitivity Analysis for Cost vs Harvesting Cost



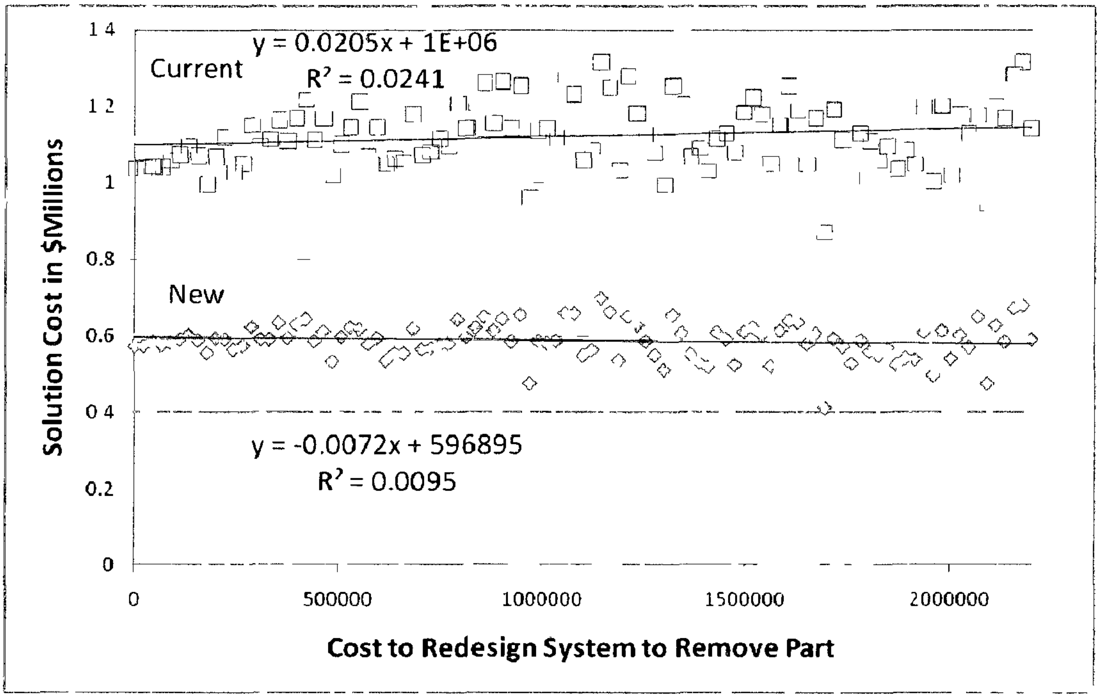
Graph 21 - Sensitivity Analysis for Cost vs. Substitution Production Gain



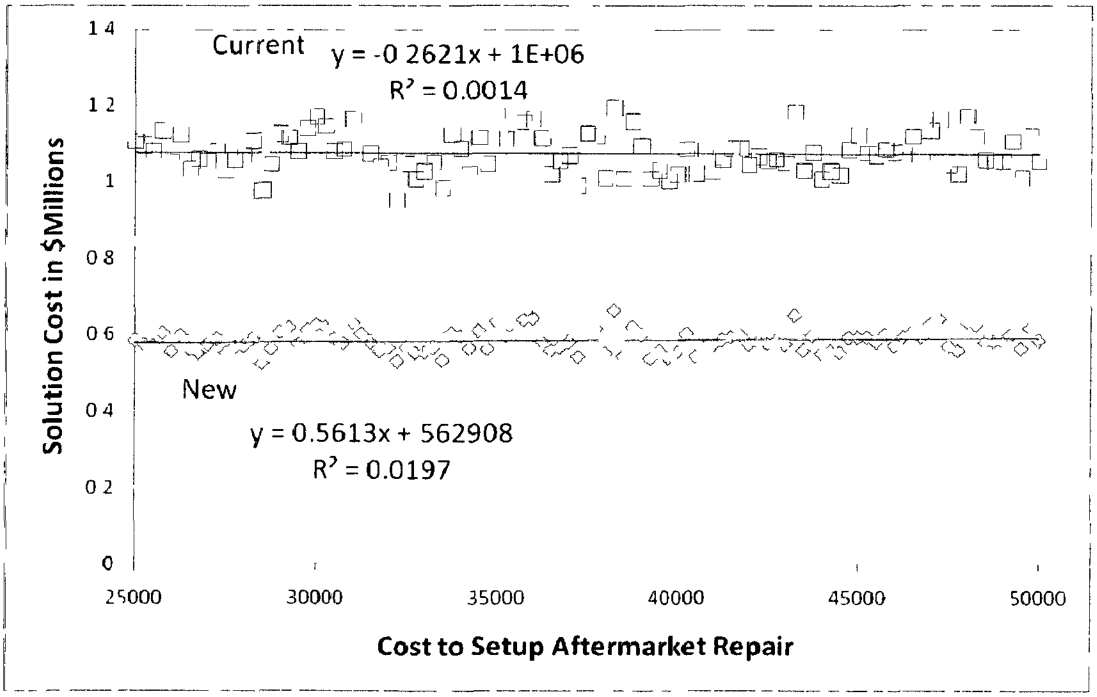
Graph 22 - Sensitivity Analysis for Cost vs. Substitution Costs



Graph 23 - Sensitivity Analysis for Cost vs. Aftermarket Production Costs



Graph 24 - Sensitivity Analysis for Cost vs. Redesign Costs



Graph 25 - Sensitivity Analysis for Cost vs. Aftermarket Setup Costs

These results are summarized in Table 7. Additionally, the coefficients that were found with the best fit line are included along with a typical value these coefficients might multiply for each variable.

	New R^2	Base R^2	New Coefficient	Base Coefficient	Typical Value
Yearly Inv Loss	0.9942	0.9957	28190	52114	5
Substitution Extension	0.0308	0.0028	792.3	524.6	3
Aftermarket Repair Setup	0.0197	0.0014	0.5613	-0.2621	30000
Redesign Cost	0.0095	0.0241	-0.0072	0.0205	500000
Aftermarket Production Setup	0.0048	0.0008	0.1206	0.0889	30000
Harvest Item Cost	0.0030	0.0203	-8.31	-39.1	250
Years to Buy	0.0010	0.0060	508.62	21.44	3
Substitution Testing Cost	0.0003	0.2029	-0.0025	0.1307	20000

Table 7 - Correlation and Coefficients from Sensitivity Analysis

By multiplying these coefficients with their corresponding typical values, a rough idea of the impact on the total cost each of these variables can be obtained. The results are summarized in Table 8.

	R^2	Impact on Cost
Yearly Inv Loss	0.9942	82.41%
Substitution Extension	0.0308	1.39%
Aftermarket Repair Setup	0.0197	9.85%
Redesign Cost	0.0095	2.10%
Aftermarket Production Setup	0.0048	2.12%
Harvest Item Cost	0.0030	1.21%
Years to Buy	0.0010	0.89%
Substitution Testing Cost	0.0003	0.03%

Table 8 - Correlation and Cost Impact from Sensitivity Analysis

The typical drop in inventory, and therefore the number of spares required, is an extremely good indicator of the total cost, having almost perfect correlation and accounting for approximately 82% of the cost. The next most relevant factor seemed to be setting up an aftermarket repair depot, which accounted for roughly 10% of the cost but is extremely dubious with a poor R^2 value of 0.02. In general, the other factors were comparably irrelevant. It is worth noting, however, that they have only proven irrelevant in the ranges and scenarios tested. In general these ranges should include any realistic scenarios, but extreme scenarios outside of those covered here could very well behave differently.

One additional observation is the large difference in the R^2 value for substitution testing costs between the FROST Method and the current method. The current method had a value of 0.2029 and the new method only 0.0003. This illustrates where much of the cost reduction comes from. Current methods result in running out of parts and having to perform a substitution so often that costs associated with substitutions have a noticeable impact on the overall cost of sustaining a system. With the FROST Method, this is not the case.

These results lend further credibility to the simulations performed to determine the cost savings in different scenarios. The results of the simulations were treated as if they were purely a function of lifetime, starting inventory levels, and number of spares required. In reality, the results are also due to the random functions used to generate parameters such the cost to setup a repair depot; had different functions and ranges been used to generate these parameters, the results would have been different. Luckily, the sensitivity analysis has demonstrated that the choice of those parameters was fairly insignificant; they do not have a relevant impact on the final cost, so even if other values had been chosen the simulations would have arrived at extremely similar results.

CHAPTER 8

RESULTS

Now that numbers have been found for both the scheduling and cost prediction portions, the cost reduction equation is ready to be used. Recall that equation 5.33 is

$$Reduction\ in\ Cost = 1 - \left(\frac{C_p}{C_c}\right)\frac{S_p + (1 + zD)\cdot(1 - S_p)}{S_c + (1 + zD)\cdot(1 - S_c)}$$

The values that get plugged into this equation are dependent on a lot of factors: lifetime, number of failures, history of failure data, starting inventory, number of parts in the system, the difference between schedules, and the spare factor. While I would like to provide a simple result for this reduction in cost equation, or a chart or lookup table that allowed someone to determine the reduction in cost simply, there are too many factors to do so. Any chart or table would be too long or complicated to use. Instead, generalities will be provided, along with a method so anyone can use the results to determine the expected cost reduction for a specific system.

First, the generalities. With every combination possible for the various simulations discussed in this paper, the Reduction in Cost ranges from -2.4% to 74.1%, with an average of 40.8%.

Many of those situations, however, are not very likely. By making a few assumptions, this range can be reduced to a more practical and likely range.

First, an assumption will be made that when choosing schedules there are a wide variety of scheduling options to choose from instead of just one or two. With so many possible schedules, it is a safe assumption that the difference between the best and second best schedule will be relatively small. This will be accounted for by limiting the results to those with a schedule difference of 1%. While other differences, such as 0.5%, are quite possible, they result in a negligible difference in cost compared to 1% and including them provides no additional value.

Second, it will be assumed that the typical unique part is not failing in the range of 1000 times per year. While some systems surely exist that are so massive that this number of parts fail, it is surely the exception and not the rule. These are the sorts of numbers that are possibly met by one or two high-failure items in a system, but not all of them. After all, with processors typically having MTBFs in the hundreds of thousands or millions of hours, a system would have to be of the size to use 10,000+ processors before it would reach this level of failures. Such systems exist, but not many.

Third, an assumption will be made that broken parts are being repaired and that the success rate is about 80%, giving a value of z=0.2.

These practical considerations provide a range which is more accurate for most systems, but for a system that does not meet these assumptions, data will be provided to find a more specific value. These considerations create a new Reduction in Cost range from 21.1% to 69.1% depending on the situation, with an average of 43.6%. These are extremely good numbers, demonstrating that the FROST Method is a huge improvement over current methods. Additionally, these results were derived by giving the current method the benefit of the doubt that the Best @ 50% method was being used. Since many programs currently use methods worse than the Best @ 50% method, those programs could expect improvements even beyond those shown in these results.

To determine more specifically the reduction in cost for any scenario, use the following method. First, determine the remaining lifetime of the system, approximately how many failures you expect each part to see a year, and how much inventory you currently have on hand. With those 3 variables in mind, use Table 9 to find a value for C_p/C_c.

Life	Failures	Starting Inventory	C_p/C_c
2	10	0	0.6199
2	100	0	0.6500
2	1000	0	0.6060
2	10	0.5	0.4374
2	100	0.5	0.6490
2	1000	0.5	0.6083
2	10	0.8	0.3100
2	100	0.8	0.7835
2	1000	0.8	0.9679
4	10	0	0.6321
4	100	0	0.6639
4	1000	0	0.6758
4	10	0.5	0.4701
4	100	0.5	0.5472
4	1000	0.5	0.5252
4	10	0.8	0.3514
4	100	0.8	0.7092
4	1000	0.8	0.7906
8	10	0	0.5992
8	100	0	0.7015
8	1000	0	0.7313
8	10	0.5	0.4824
8	100	0.5	0.4817
8	1000	0.5	0.5139
8	10	0.8	0.4074
8	100	0.8	0.6003
8	1000	0.8	0.6366
16	10	0	0.6330
16	100	0	0.7630
16	1000	0	0.7749
16	10	0.5	0.4275
16	100	0.5	0.5213
16	1000	0.5	0.5708
16	10	0.8	0.4980
16	100	0.8	0.5107
16	1000	0.8	0.5069

Table 9 - Parameter-Based Cost Ratios

Next, determine the remaining lifetime of the system, the number of failures your typical MTBF data is based on (V = vendor data), the number of unique parts in your system, and an estimate of the difference, D, in percentage between schedules you might compare. Use these parameters to find values for S_c and S_p in Table 10.

MTBF				S_c							S_p						
Base	Life	Parts	D>	0.5	1	2	5	10	25	100	0.5	1	2	5	10	25	100
V	2	5		0.648	0.598	0.636	0.634	0.626	0.606	0.624	0.480	0.496	0.506	0.536	0.516	0.592	0.716
V	4	5		0.638	0.634	0.600	0.640	0.618	0.628	0.644	0.488	0.536	0.506	0.520	0.542	0.606	0.770
V	8	5		0.606	0.638	0.648	0.626	0.654	0.638	0.656	0.494	0.492	0.530	0.564	0.588	0.662	0.854
V	16	5		0.636	0.642	0.648	0.664	0.630	0.632	0.638	0.578	0.518	0.534	0.554	0.618	0.678	0.904
V	32	5		0.610	0.658	0.632	0.630	0.618	0.604	0.652	0.510	0.522	0.546	0.572	0.584	0.730	0.910
V	2	25		0.640	0.606	0.576	0.568	0.588	0.596	0.594	0.534	0.534	0.522	0.550	0.520	0.626	0.728
V	4	25		0.612	0.572	0.540	0.604	0.604	0.572	0.622	0.494	0.488	0.556	0.532	0.614	0.662	0.834
V	8	25		0.608	0.586	0.588	0.532	0.604	0.602	0.574	0.502	0.538	0.508	0.526	0.582	0.714	0.886
V	16	25		0.612	0.584	0.588	0.578	0.560	0.584	0.574	0.512	0.512	0.502	0.564	0.602	0.730	0.902
V	32	25		0.628	0.598	0.566	0.586	0.576	0.584	0.562	0.528	0.534	0.530	0.560	0.604	0.744	0.924
V	2	100		0.500	0.514	0.530	0.476	0.488	0.554	0.500	0.508	0.522	0.528	0.544	0.588	0.612	0.752
V	4	100		0.520	0.522	0.508	0.502	0.524	0.488	0.514	0.502	0.522	0.504	0.520	0.594	0.686	0.900
V	8	100		0.510	0.512	0.500	0.532	0.506	0.578	0.514	0.508	0.502	0.504	0.602	0.626	0.720	0.948
V	16	100		0.540	0.502	0.492	0.522	0.490	0.534	0.520	0.510	0.542	0.546	0.528	0.670	0.770	0.930
V	32	100		0.540	0.512	0.460	0.486	0.492	0.524	0.478	0.476	0.516	0.516	0.588	0.624	0.788	0.956
1	2	5		0.656	0.660	0.646	0.640	0.638	0.616	0.618	0.498	0.488	0.524	0.522	0.520	0.536	0.634
1	4	5		0.642	0.658	0.622	0.642	0.644	0.650	0.658	0.502	0.494	0.482	0.542	0.512	0.578	0.648
1	8	5		0.624	0.624	0.616	0.680	0.674	0.624	0.620	0.492	0.494	0.520	0.522	0.512	0.572	0.708
1	16	5		0.592	0.682	0.646	0.628	0.636	0.640	0.628	0.486	0.496	0.494	0.508	0.544	0.576	0.676
1	32	5		0.630	0.636	0.690	0.648	0.664	0.626	0.610	0.510	0.542	0.522	0.500	0.510	0.578	0.694
1	2	25		0.612	0.596	0.600	0.608	0.622	0.588	0.618	0.470	0.508	0.516	0.534	0.532	0.524	0.656
1	4	25		0.622	0.620	0.582	0.594	0.612	0.638	0.646	0.514	0.526	0.528	0.570	0.520	0.580	0.654
1	8	25		0.626	0.600	0.558	0.610	0.638	0.606	0.594	0.536	0.514	0.512	0.512	0.524	0.574	0.730
1	16	25		0.602	0.552	0.608	0.608	0.608	0.610	0.650	0.520	0.510	0.484	0.530	0.574	0.576	0.742
1	32	25		0.616	0.592	0.572	0.650	0.568	0.594	0.614	0.518	0.482	0.518	0.512	0.518	0.570	0.762

Table 10 - Parameter-Based Success Rates for Choosing Better Schedule

MTBF				S_c							S_p						
Base	Life	Parts	D>	0.5	1	2	5	10	25	100	0.5	1	2	5	10	25	100
1	2	100		0.550	0.586	0.532	0.554	0.534	0.614	0.580	0.508	0.456	0.498	0.536	0.524	0.544	0.694
1	4	100		0.520	0.524	0.560	0.558	0.560	0.562	0.574	0.508	0.568	0.500	0.520	0.538	0.560	0.742
1	8	100		0.530	0.552	0.554	0.536	0.600	0.588	0.568	0.486	0.500	0.490	0.542	0.494	0.584	0.762
1	16	100		0.528	0.520	0.558	0.582	0.516	0.540	0.554	0.490	0.486	0.490	0.536	0.572	0.614	0.810
1	32	100		0.560	0.558	0.584	0.590	0.572	0.526	0.568	0.504	0.498	0.494	0.530	0.556	0.630	0.788
2	2	5		0.638	0.626	0.614	0.626	0.638	0.610	0.600	0.470	0.516	0.524	0.518	0.556	0.562	0.650
2	4	5		0.666	0.600	0.632	0.606	0.660	0.640	0.632	0.512	0.532	0.484	0.516	0.470	0.570	0.758
2	8	5		0.596	0.628	0.634	0.658	0.578	0.658	0.588	0.524	0.500	0.488	0.482	0.558	0.622	0.778
2	16	5		0.640	0.622	0.620	0.628	0.626	0.622	0.600	0.498	0.554	0.536	0.542	0.550	0.644	0.800
2	32	5		0.628	0.630	0.622	0.654	0.624	0.662	0.634	0.506	0.474	0.446	0.556	0.524	0.620	0.778
2	2	25		0.550	0.554	0.602	0.606	0.576	0.566	0.610	0.494	0.528	0.512	0.530	0.548	0.594	0.712
2	4	25		0.612	0.636	0.592	0.598	0.588	0.622	0.624	0.474	0.514	0.498	0.536	0.572	0.642	0.792
2	8	25		0.596	0.606	0.578	0.562	0.630	0.616	0.598	0.472	0.528	0.534	0.524	0.558	0.678	0.850
2	16	25		0.618	0.604	0.590	0.576	0.596	0.602	0.618	0.488	0.500	0.520	0.526	0.572	0.618	0.850
2	32	25		0.582	0.582	0.586	0.582	0.568	0.572	0.604	0.500	0.474	0.508	0.546	0.566	0.656	0.848
2	2	100		0.544	0.484	0.494	0.554	0.544	0.508	0.562	0.522	0.510	0.524	0.516	0.546	0.592	0.762
2	4	100		0.526	0.530	0.536	0.542	0.522	0.550	0.540	0.514	0.508	0.486	0.516	0.574	0.660	0.820
2	8	100		0.574	0.548	0.556	0.556	0.508	0.508	0.492	0.452	0.514	0.492	0.494	0.592	0.656	0.896
2	16	100		0.560	0.480	0.548	0.564	0.552	0.528	0.500	0.512	0.540	0.490	0.540	0.596	0.676	0.916
2	32	100		0.552	0.570	0.502	0.562	0.540	0.496	0.526	0.496	0.512	0.526	0.528	0.610	0.702	0.938
4	2	5		0.628	0.644	0.648	0.626	0.622	0.620	0.624	0.536	0.494	0.480	0.512	0.518	0.594	0.694
4	4	5		0.628	0.612	0.626	0.638	0.642	0.644	0.648	0.494	0.524	0.504	0.530	0.530	0.636	0.822
4	8	5		0.646	0.636	0.642	0.648	0.602	0.592	0.632	0.504	0.472	0.530	0.560	0.546	0.636	0.854
4	16	5		0.602	0.594	0.628	0.644	0.604	0.630	0.626	0.496	0.470	0.548	0.540	0.586	0.662	0.870
4	32	5		0.632	0.644	0.652	0.608	0.626	0.622	0.626	0.500	0.514	0.542	0.526	0.576	0.642	0.884

Table 10 Continued

MTBF				S_c							S_p						
Base	**Life**	**Parts**	**D>**	**0.5**	**1**	**2**	**5**	**10**	**25**	**100**	**0.5**	**1**	**2**	**5**	**10**	**25**	**100**
4	2	25		0.586	0.594	0.566	0.586	0.542	0.550	0.592	0.540	0.504	0.534	0.506	0.544	0.592	0.744
4	4	25		0.566	0.574	0.586	0.586	0.582	0.570	0.612	0.460	0.518	0.540	0.484	0.606	0.662	0.868
4	8	25		0.618	0.610	0.616	0.620	0.578	0.596	0.568	0.514	0.526	0.544	0.520	0.570	0.716	0.916
4	16	25		0.634	0.596	0.598	0.586	0.544	0.610	0.612	0.486	0.504	0.494	0.568	0.594	0.714	0.956
4	32	25		0.622	0.584	0.592	0.596	0.560	0.574	0.606	0.518	0.520	0.524	0.550	0.640	0.740	0.972
4	2	100		0.540	0.516	0.488	0.526	0.524	0.512	0.504	0.460	0.470	0.526	0.514	0.546	0.598	0.794
4	4	100		0.534	0.516	0.472	0.504	0.496	0.526	0.506	0.496	0.502	0.532	0.560	0.582	0.714	0.942
4	8	100		0.492	0.556	0.530	0.496	0.526	0.538	0.548	0.502	0.514	0.540	0.590	0.628	0.780	0.966
4	16	100		0.510	0.548	0.482	0.504	0.518	0.510	0.524	0.488	0.464	0.528	0.550	0.646	0.802	0.984
4	32	100		0.530	0.506	0.498	0.540	0.542	0.548	0.530	0.530	0.552	0.520	0.582	0.682	0.844	0.992
8	2	5		0.668	0.628	0.620	0.640	0.614	0.646	0.642	0.500	0.516	0.526	0.538	0.558	0.576	0.740
8	4	5		0.630	0.660	0.648	0.648	0.656	0.624	0.670	0.524	0.502	0.522	0.532	0.568	0.656	0.854
8	8	5		0.622	0.620	0.618	0.644	0.648	0.632	0.636	0.514	0.520	0.506	0.540	0.592	0.662	0.872
8	16	5		0.636	0.606	0.682	0.628	0.610	0.630	0.622	0.498	0.490	0.530	0.550	0.574	0.700	0.902
8	32	5		0.652	0.640	0.650	0.626	0.622	0.632	0.674	0.508	0.518	0.506	0.530	0.592	0.686	0.928
8	2	25		0.614	0.568	0.592	0.586	0.566	0.580	0.632	0.514	0.502	0.506	0.516	0.514	0.608	0.762
8	4	25		0.576	0.644	0.600	0.608	0.590	0.578	0.580	0.496	0.516	0.508	0.558	0.602	0.686	0.912
8	8	25		0.564	0.604	0.620	0.576	0.596	0.598	0.612	0.514	0.500	0.528	0.544	0.608	0.730	0.954
8	16	25		0.590	0.544	0.580	0.566	0.574	0.580	0.614	0.538	0.528	0.532	0.642	0.598	0.806	0.982
8	32	25		0.618	0.520	0.594	0.602	0.566	0.614	0.578	0.504	0.546	0.502	0.570	0.638	0.810	0.974
8	2	100		0.518	0.528	0.520	0.516	0.510	0.556	0.500	0.512	0.506	0.512	0.532	0.594	0.624	0.804
8	4	100		0.546	0.516	0.518	0.538	0.444	0.546	0.510	0.514	0.554	0.578	0.528	0.604	0.756	0.930
8	8	100		0.490	0.502	0.478	0.476	0.490	0.516	0.472	0.484	0.546	0.536	0.624	0.680	0.824	0.966
8	16	100		0.508	0.486	0.488	0.472	0.522	0.476	0.514	0.502	0.556	0.558	0.640	0.710	0.886	0.996
8	32	100		0.508	0.534	0.548	0.496	0.488	0.512	0.510	0.486	0.540	0.560	0.610	0.754	0.904	1.000

Table 10 Continued

Last, determine your spare factor, z. If you need a spare to replace every single part that fails, use z=1. If you successfully repair, harvest, or otherwise replace 80% of parts and only need a spare to replace the other 20%, use z=0.2.

This should leave you with all the values you need for the Reduction in Cost equation, equation 5.33.

$$Reduction\ in\ Cost = 1 - \left(\frac{C_p}{C_c}\right)\frac{S_p + (1 + zD)\cdot(1 - S_p)}{S_c + (1 + zD)\cdot(1 - S_c)}$$

Here is one example of how to determine the expected cost reduction. Imagine you are sustaining a system with an 8 year lifetime where you typically expect less than 10 failures per part per year, and you have no inventory on hand. This would give you a C_p/C_c of 0.5992.

You also have no historical failure data and will be relying on vendor estimates, your system contains 100 unique parts, and you have a large number of schedules to choose from so you expect the difference between the best to schedules to be approximately 1%. Using this information, you find a S_p of 0.502 and an S_c of 0.512.

Lastly, you plan to set up a quality repair process which successfully repairs 90% of parts, leaving you with a spare factor of 0.1.

Plugging these values into the equation, you find that, in comparison with current methods, you can expect a reduction in variable sustainment costs of approximately 40%.

CHAPTER 9

ADAPTING THE MODEL FOR NON-ELECTRONIC PARTS

The method provided so far has been aimed solely at solving sustainment for electronics. Electronics are easier to deal with in such a method because they essentially have a time-independent failure rate which causes the mathematics behind their failures to be comparatively simple (Wilkins, 2002). However, this method can easily be adapted to handle non-electronic parts as well.

There are two issues that need to be addressed to adapt this method. First, the provided distributions were all created from sample populations which only included electronic parts. If these same distributions were applied to a model for non-electronic parts, the correlation would likely be poor. Anyone wanting to adapt this method should first gather appropriate data samples and perform distribution fitting to create new distributions for variables such as time to end of production.

The second issue is that that gamma function used in my method to predict failures is inappropriate for parts with time-dependent failure rates. A different method is required to predict the number of failures that occur at any given time. I will provide this method so that this model can easily be adapted beyond electronic parts.

Time-dependent failure rates of mechanical parts are generally modeled with a Weibull distribution (Todinov, 2005). The Weibull distribution includes shape and scale parameters which allow it to be customized so that failures either increase or decrease over time. This allows it to be used to model the three sections of the bathtub curve.

During the infant mortality region, birth defects in parts cause them to fail. Most serious defects will cause a failure very early in the life of a part. As time passes and a part continues to function, it becomes more and more likely that the part does not contain any fatal defects, and the probability of failure decreases (Wilkins, 2002).

During the stable region, random chance causes some event to break the part. This is equally likely to occur at any given time, so the failure rate stays constant.

During the wear-out region, the part starts to break down. As time passes, it becomes more likely that it will fail.

These three regions combine to form the bathtub curve for a part, describing its likelihood of failure at any given time. Note that these regions can and generally do overlap. Though rare, it is possible for a part to break due to a birth defect well into the region of time you would normally expect failures to be caused by wear-out.

The CDF for a single Weibull distribution is (Todinov, 2005)

$$F(T) = 1 - e^{-\left(\frac{T-t_0}{\eta}\right)^{\beta}} \tag{9.1}$$

where

T = time to failure,

t_0 = time when $F(T) = 0$,

η = scale parameter,

β = shape parameter.

Each of the three regions of failure has its own Weibull Distribution. For the infant mortality region, β<1 because the probability of failure decreases over time. Additionally, $t_0 = 0$ because infant mortality starts from the first moment. For the stable region, β =1 because the probability of failures is stable over time, and $t_0 = 0$ because there is always a probability of a random event. For the wear-out region, β>1 because the probability of failures is increasing over time.

These can be combined simply using basic knowledge of PDFs. Imagine a distribution being created based on a sample of size N, with S different modes of failure. Each of those modes of failure occurs N_i times, where i represents a mode. This means that the fraction of total failures caused by mode i is equal to N_i/N. This fact can be used to combine PDFs as follows.

$$f(T) = \sum_{i=1}^{S} \frac{N_i}{N} f_i(T) \tag{9.2}$$

Now that PDFs have been provided, this knowledge of combining PDFs can be used to create a single mixed Weibull Distribution. Starting off with the relationship between CDFs and PDFs,

$$F(T) = \int f(t)dt \tag{9.3}$$

$$F(T) = \frac{1}{N}\sum_{i=1}^{S} N_i \int f_i(T)\,dt \tag{9.4}$$

$$F(T) = \frac{1}{N}\sum_{i=1}^{S} N_i \int \frac{dF_i(T)}{dt}\,dt \tag{9.6}$$

$$N \cdot F(T) = \sum_{i=1}^{S} N_i\, F_i(T) \tag{9.7}$$

Using the previously listed CDF for the Weibull distribution and the knowledge of values for t_0 and β in different regions, this equation can be used to create the CDF for failure rates for non-electronic parts. Define the infant mortality region as i=1, the stable region as i=2, and the wear-out region as i=3, leaving the following equation.

$$N \cdot F(T) = \tag{9.8}$$
$$N_1\left(1 - exp\left[-\left(\frac{T}{\eta_1}\right)^{\beta_1}\right]\right) + N_2\left(1 - exp\left[-\left(\frac{T}{\eta_2}\right)\right]\right) + N_3\left(1 - exp\left[-\left(\frac{T-t_{0_3}}{\eta_3}\right)^{\beta_3}\right]\right)$$

which is equal to

$$N_1 + N_2 + N_3 - N \cdot F(T) \tag{9.9}$$
$$= N_1 exp\left[-\left(\frac{T}{\eta_1}\right)^{\beta_1}\right] + N_2 exp\left[-\left(\frac{T}{\eta_2}\right)\right] + N_3 exp\left[-\left(\frac{T-t_{0_3}}{\eta_3}\right)^{\beta_3}\right]$$

Since by definition $N_1+N_2+N_3=N$, the left side of this equation can be further simplified to N(1-F(T)). At this point the fact that by definition the CDF is a probability from 0 to 1 can be used. Because each probability in that range is equally likely, this equation can be used to generate a random variable by substituting in a uniformly distributed random variable from 0 to 1, denoted as U, for the probability. This makes the left side of the equation N(1-U). A useful property of a uniformly distributed random variable from 0 to 1 is that 1-U=U. This gives a final equation of

$$U = \frac{N_1}{N} exp\left[-\left(\frac{T}{\eta_1}\right)^{\beta_1}\right] + \frac{N_2}{N} exp\left[-\left(\frac{T}{\eta_2}\right)\right] + \frac{N_3}{N} exp\left[-\left(\frac{T-t_{03}}{\eta_3}\right)^{\beta_3}\right] \tag{9.10}$$

Note that this is only correct for values of T larger than t_{03}. By plugging in appropriate values for the variables in this equation, this can be solved for T, which tells the time at which a single part will fail. For a large population of items, this can be calculated multiple times, once for each item, to create a single run demonstrating when items will fail. This implementation does not include replacements and would need to be adjusted to include them, possibly by calculating the equation again starting at the time of replacement.

Programmatically, making this adjustment to the proposed method is relatively simple. However, the number of variables that need to be acquired for each part has increased significantly. With electronic parts, there were only two variables in this part of the calculation: the number of failures, k, and the time used, T. With the non-electronic implementation this increases significantly, to 10 variables: N_1, N_2, N_3, N, β_1, β_3, t_{03}, η_1, η_2, and η_3. If appropriate values can be found for these variables, however, this method can easily be adapted to include all systems instead of only electronic ones.

CHAPTER 10

CONCLUSION

Overall, the FROST Method has shown itself to be a significant improvement over current methods. For a realistic range of system parameters, the reduction in cost can be expected to be somewhere in the range of 21.1% to 69.1% depending on the situation, with an average of 43.6%. The majority of this reduction comes from making better choices for individual sustainment solutions, and a small amount from making better scheduling choices.

This method is able to make these better choices by considering consequences using a Monte Carlo process. A review of current products and research showed that this consideration of consequences is not currently being done. In order to be more efficient, managers need to base their solutions not on how many spare parts they think they need, but on what the consequence will be if they run out of those spares. This paradigm shift leads to some very different solutions, and those different solutions can lead to approximately a 43.6% decrease in the money spent on inventory and logistical/engineering solutions for the sustainment of electronic systems. In fact, the real cost reduction can generally be expected to surpass this number since it is based on a comparison against the best-case scenario for the current method instead of the typical scenario.

Implementing this method does require keeping track of a good deal more variables than are required for current methods, so this method does impose an additional burden. An equation has been provided which allows anyone to make an approximation of the savings they could expect to see if they implemented this method. Those savings can be weighed against this additional burden to determine whether this method is worthwhile for any given system.

The additional burden imposed by this method will, for a typical system, be able to be handled by a single employee. Since this method is expected to save 40+%, in general as long a single employee represents less than 40% of the cost to sustain a system, this will be a profitable method to apply. This would mean the addition of a third person to a two person team could be expected to improve efficiency if it allowed the implementation of the proposed method.

The results in this dissertation are also dependent on a number of assumptions which are listed in the appendix. These assumptions need to hold true for this method to be applicable. For most electronics-based systems this list of assumptions will hold true.

Before this method is applied to any system, the equations and assumptions provided should be evaluated to ensure it is worthwhile. However, with the above considerations in mind, a general rule would be that this method can be expected to improve efficiency for any electronics-based system which is currently large enough to involve at least two people working to sustain it.

REFERENCES

Ashby, W. R. (1958). *An Introduction to Cybernetics.* New York: Wiley.

Barkenhagen, M. E. (2003). The Sunset Supply Base: Long Term COTS Supportability, Implementing Affordable Methods and Processes. *Thesis, US Naval Postgraduate School* .

Beck, D. S. (2003). Microelectronic Obsolescence Management. *Thesis, US Naval Postgraduate School* .

Bevilacqua, M., & Braglia, M. (2000). The analytic hierarchy process applied to maintenance strategy selection. *Reliability Engineering & System Safety* , 71-83.

Center for Environmental Health. (2010, May 12). *Healthcard Giants Endorse Strong Environmental Standards for Electronic Equipment.* Retrieved December 2010, from CEH.org Press Releases: http://www.ceh.org/index.php?option=com_content&task=view&id=438&Itemid=166

Chief of Naval Operations. (1999). *OPNAVINST 4441.12C Retail Supply Support of Naval Activities and Operating Forces.* Washington DC: Department of the Navy.

Clemson, B. (1991). *Cybernetics: A New Management Tool.* Routledge.

DoD. (1995). *MIL-HDBK-217F (NOTICE 2), MILITARY HANDBOOK: RELIABILITY PREDICTION OF ELECTRONIC EQUIPMENT.* Washington DC: Department of Defense.

Feng, D. S. (2007). Lifetime Buy Optimization to Minimize Lifecycle Cost. *Proceedings of the 2007 Aging Aircraft Conference.*

Frangopol, D. L. (1997, October). Life-Cycle Cost Design of Deteriorating Structures. *Journal of Structural Engineering* , 1390-1401.

Hamilton, & Chin. (2001, March). Aging Military Electronics: What Can the Pentagon Do? *National Defense Magazine* .

Herald, T. E., Verma, D., & Lechler, T. (2007). A Model Proposal to Forecast System Baseline Evolution due to Obsolescence through System Operation. *Conference on Systems Engineering Research 2007 Proceedings.* Hoboken: Stevens Institute of Technology.

Howard, M. A. (2002). Component Obsolescence: It's Not Just for Electronics Anymore. *Proceedings of the Diminishing Manufacturing and Material Shortages (DMSMS) 2002 Conference.*

Keller, J. (2006, March). Defense spending set to increase for electronics and electro-optics programs in 2007. *Military & Aerospace Electronics* .

Miser, H., & Quade, E. (1985). *Handbook of Systems Analysis, Vol 1.* Elsevier Science.

Mummolo, G. R. (2008). *A Fuzzy Approach for Medical Equipment Replacement Planning.*

Naval Sea Systems Command. (2009). *Naval Sea Systems Command Strategic Business Plan 2009-2013.* Washington DC: Naval Sea Systems Command.

Perry, W. (1994). *Memorandum from the Secretary of Defense to the Secretaries of the Military Departments, "Specifications & standards -- A new way of doing business".* The Pentagon, Office of the Secretary of Defense.

Rugina, D. (2000). *Life Time Buy Modeling at Agilent Technologies.* University of British Columbia.

Shachtman, N. (2010, August 31). *HP Holds Navy Network 'Hostage' for $3.3 Billion.* Retrieved from WIRED: http://www.wired.com/dangerroom/2010/08/hp-holds-navy-network-hostage/

Singh, P., Sandborn, P., Lorenson, D., & Geiser, T. (2002). Determining Optimum Redesign Plans for Avionics Considering Electronic Part Obsolescence Forecasts. *Proceedings of the World Aviation Congress 2002.* Phoenix.

Skyttner, L. (2001). *General Systems Theory.* Singapore: World Scientific.

Thissen, W. A., & Twaalfhoven, P. G. (2001). Toward a Conceptual Structure for Evaluating Policy Analytic Activities. *European Journal of Operational Research , 129* (3).

Todinov, M. (2005). *Reliability and Risk Models.* Chichester, UK: John Wiley & Sons.

Wang, L. C. (2007). Selection of Optimum Maintenance Strategies Based on a Fuzzy Analytic Hierarchy Process. *International Journal of Production Economics* , 151-163.

Weisstein, E. W. (2011, March 26). *Gamma Distribution.* Retrieved from MathWorld--A Wolfram Web Resource: http://mathworld.wolfram.com/GammaDistribution.html

Weisstein, E. W. (2011, Mar 26). *Incomplete Gamma Function.* Retrieved from MathWorld--A Wolfram Web Resource.

Weisstein, E. W. (2011, March 26). *Regularized Gamma Function.* Retrieved from MathWorld--A Wolfram Web Resource: http://mathworld.wolfram.com/RegularizedGammaFunction.html

Wilkins, D. J. (2002, December). *The Bathtub Curve and Product Failure Behavior.* (R. HotWire, Ed.) Retrieved from Weibull.com: http://www.weibull.com/hotwire/issue22/hottopics22.htm

APPENDICES

APPENDIX A: MATHEMATICS SUMMARY

A.1 Indexes

E = Equipment Index

This represents a particular entry in a hypothetical one-dimensional array of all equipment. It is used as an index for other arrays. For example, $U_{E,L}$ might represent the Usage Factor for a specific piece of equipment in a specific building.

L = Location Index

This represents a particular entry in a hypothetical one-dimensional array of all locations where powered equipment might be located, such as buildings or ships. It is used as an index for other arrays. For example, $U_{E,L}$ might represent the Usage Factor for a specific piece of equipment in a specific building.

p = Part Index

This represents a particular entry in a hypothetical one-dimensional array of all parts. It represents a specific part being analyzed and is generally used as an index for other arrays. For example, G_p represents the value of array G for a particular part, p.

t = Time Index

This represents a particular time in a hypothetical one-dimensional array of time, and is used as an index for other arrays. For example, $F_{p,t}$ might represent the number of failures that occurred for part p during month t.

A.2 Parameters and Inputs

$D_p(b)$ = Buy Delay

This represents the delay that occurs between a part p failing and a new part being purchased as a replacement and put into inventory.

$D_p(s)$ = Support Delay

This represents the delay that occurs between a part p failing and that part being returned to inventory after being repaired.

$J_{p,L,E}$ = Harvest Flag

This represents whether a particular part p is able to be harvested out of equipment E at location L. It is true or false, and assigned a value of 1 or 0.

K = Confidence Level

This represents the confidence level at which analyses are being performed. A value of 0.5 would mean results are equally likely to be better or worse than outputted. A value of 0.95 would mean results have a 5% chance of being worse than outputted.

$M_p(s)$ = End of Support

This represents the date when repairs are no long available for part p.

$M_p(b)$ = End of Buy

This represents the date when purchasing additional parts is no longer an option for part p.

$Q_{E,L,t}(E)$ = Quantity of Equipment per Location

This represents the quantity of equipment E at location L during time t.

$Q_{p,E}(p)$ = Quantity of Parts per Equipment

This represents the quantity of part p which exist in each piece of equipment E.

$R_p(h)$ = Harvest Rate

This represents the fraction of parts that are successfully harvested from a system and returned to inventory in working order for each instance of part p that is removed from a location.

$R_p(s)$ = Support Rate

This represents the fraction of parts that are successfully repaired for failures of part p while repair is still available.

$U_{E,L}$ = Usage Factor

This represents the fraction of the time that a piece of equipment E is powered on and parts are actively failing, for a particular location L.

v_p = Repair Flag

This is a variable representing whether or not part p is repairable. 1 means repairable, 0 means not repairable.

Θ_p = MTBF

This represents the mean time between failures for part p.

A.3 Variables

$A_{p,t}$ = Replacement Array

This is a two-dimensional array indicating the number of failures for part p that will be replaced, either through repair or purchase, during time t.

$B_{p,t}$ = Buy Array

This is a two-dimensional array indicating the rate at which failed parts p are replaced by new purchases at time t.

$C_{p,t}(a)$ = Actual Parts in Fleet

This is a two-dimensional array indicating the total parts of type p being considered in the fleet at time t. This is used for accounting and harvesting but not for calculating failures.

$C_{p,t}(l)$ = Live Parts in Service

This is a two-dimensional array indicating the total parts of type p being considered which are actively powered and failing at time t. This is actually used for calculating failures, and includes adjustments for parts that may be deployed and harvestable but not powered on or failing.

$F_{p,t}$ = Failure Array

This is a two-dimensional array indicating the number of failures for part p that will occur during time t.

G_p = Failures per Usage

This is a one-dimensional array indicating the number of failures expected to occur if a single instance of part p is powered on for one time period, Δt.

$H_{p,t}$ = Harvest Array

This is a two-dimensional array indicating the quantity of part p successfully harvested and returned to inventory in working order during time t. This opportunity occurs when a piece of equipment is removed from a location.

$I_{p,t}$ = Inventory Array

This is a two-dimensional array indicating the amount of spare inventory for part p that will be available during time t.

$S_{p,t}$ = Support Array

This is a two-dimensional array indicating the rate at which failed parts p are repaired at time t.

A.4 Equations

$$I_{p,t} = I_{p,t-1} - F_{p,t-1} + A_{p,t-1} + H_{p,t-1}$$

$$\Gamma inv(\alpha_p, K, 1) = solve\ [Gamma(\alpha_p, x) = K\ ]\ for\ x$$

$$C_{p,t}(a) = \sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t}(E)$$

$$C_{p,t}(l) = \sum_L \sum_E Q_{p,E}(p) \cdot Q_{E,L,t}(E) \cdot U_{E,L}$$

$$F_{p,t} = C_{p,t}(l) \cdot G_p$$

$$G_p = \frac{\Gamma inv(\alpha_p, K, 1)}{\sum_t \ C_{p,t}(l)}$$

$$\alpha_p = \sum_t (C_{p,t}(l) \cdot \Delta t / \theta_p)$$

$$H_{p,t} = \frac{R_p(h)}{2} \sum_L \sum_E \Big[\Big(\Big|Q_{p,E}(p) \cdot \big(Q_{E,L,t}(E) - Q_{E,L,t-1}(E)\big)\Big| - Q_{p,E}(p) \cdot \big(Q_{E,L,t}(E) - Q_{E,L,t-1}(E)\big)\Big) \cdot J_{p,L,E}\Big]$$

$$A_{p,t} = F_{p,t-D_p(s)} \cdot S_{p,t-D_p(s)} + F_{p,t-D_p(b)} \cdot B_{p,t-D_p(b)}$$

$$B_{p,t-D_p(b)} = 0$$

$$S_{p,t} = \begin{cases} R_p(s) \cdot V_p, & t < \mathrm{M_p(s)} \\ 0, & t \geq \mathrm{M_p(s)} \end{cases}$$

$$Gamma(\alpha_p, x) = \begin{cases} \dfrac{x^{\alpha_p - 1} e^{-x}}{\Gamma(\alpha_p)}, & when\ using\ Monte\ Carlo \\ \dfrac{\gamma(\alpha_p, x)}{\Gamma(\alpha_p)}, & when\ using\ Single\ Calculation \end{cases}$$

$$\gamma\left(\alpha_p, x\right) = \int_0^x t^{\alpha_p - 1} e^{-t} dt$$

APPENDIX B: DISTRIBUTIONS

B.1 Real MTBF based on history

$$\theta = \frac{2T}{\chi^2_{\alpha,2k}}$$

Where T is the time of use witnessed for the part, k is the number of failures witnessed for the part, and the denominator represents a Chi-Square distribution for probability α with 2k degrees of freedom.

B.2 PDF for Witnessed MTBF / Vendor-Claimed MTBF

$$f(x) = \frac{\alpha k \left(\frac{x}{\beta}\right)^{\alpha-1}}{\beta\left(1 + \left(\frac{x}{\beta}\right)^{\alpha}\right)^{k+1}}$$

k=2.4483

α=0.85988

β=7.6449

B.3 PDF for End of Production / Vendor-Claimed End of Production

Select value Y from uniformly distributed random variable from 0 to 1.

If Y < 0.01519, use f(x) = 0

If Y > 0.44810, use f(x) = 1

Else use

$$f(x) = \frac{\alpha k \left(\frac{x}{\beta}\right)^{\alpha-1}}{\beta\left(1 + \left(\frac{x}{\beta}\right)^{\alpha}\right)^{k+1}}$$

k=1.3081

α=1.7704

β=1.6636

B.4 PDF for End of Support / Vendor-Claimed End of Support

Select value Y from uniformly distributed random variable from 0 to 1.

If Y < 0.05310, use f(x) = 0

If Y > 0.51880, use f(x) = 1

Else use

$$f(x) = \frac{1}{\sigma}\exp\left(-(1 + kz)^{-\frac{1}{k}}\right)(1 + kz)^{-1-1/k}$$

where z=(x-μ)/σ.

k=0.53333

σ=0.32944

μ=0.34143

B.5 PDF for Repair Rate

$$f(x) = \alpha_1 \alpha_2 x^{\alpha_1 - 1}(1 - x^{\alpha_1})^{\alpha_2 - 1}$$

α_1=2.2636

α_2=0.14035

APPENDIX C: ASSUMPTIONS

The following assumptions were used in the creation of the proposed method. Caution should be used in applying this model to a system where these assumptions do not hold true.

- The goal is to maintain 100% system health as efficiently as possible
- Problems with system health can be identified entirely by determining when a part will fail and no inventory will be available to immediately replace it.
- Sufficient data will be provided to the model for it to operate.
- Schedules provided to the model are mistake free.
- The sample size will be set sufficiently high to create repeatable results.
- The model is used by engineers and logisticians to determine when they should take action.
- Relevant future costs can be based entirely on inventory level, part type, date, and other variables which are part of the model. Future costs which cannot be determined solely by these variables are irrelevant.
- The viability of solutions can be accurately determined using simple logic.
- When a new solution type is discovered, it will be added to the model before it starts being used.
- Failure rates for electronic parts are age-independent.

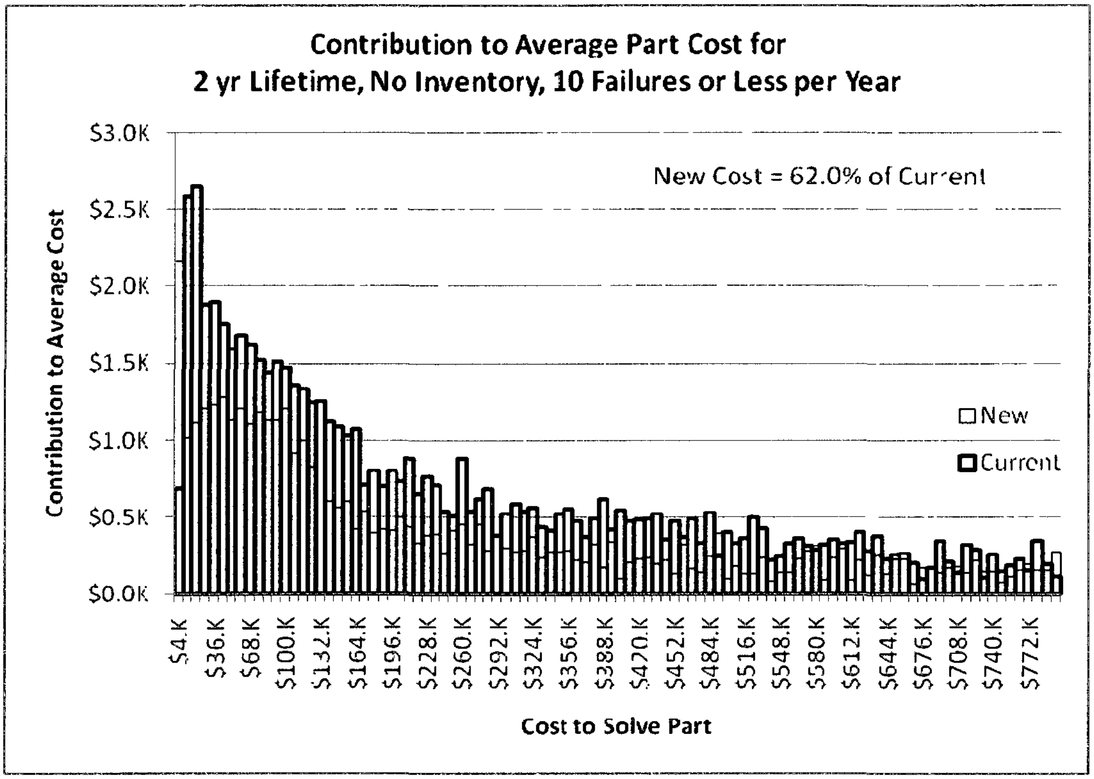
APPENDIX D: EXAMPLE CODE TO DETERMINE WHICH SOLUTIONS WILL BE USED BASED ON COST AND SUFFICIENCY

```
'== THE FOLLOWING FUNCTIONS DETERMINE WHETHER A SOLUTION WILL BE USED.
'== 0 = not used
'== 1 = always used
'== 2 = possibly used later depending on what happens in the future
'== AvgSolutionOrder(x) returns the name of the (x+1)th cheapest solution

Function IsBuyUsed() As Integer
  If (Not BuyOption) Then
    Return 0
  ElseIf StockIsSufficient Then
    Return 0
  ElseIf (SolutionCostEstimate("Buy") > SolutionCostEstimate("SSB")) Or (SolutionCostEstimate("Buy") >_
  SolutionCostEstimate("Redesign")) Or (SolutionCostEstimate("Buy") > SolutionCostEstimate("Substitution")) Then
    Return 0
  ElseIf ((SolutionCostEstimate("Buy") > SolutionCostEstimate("Harvest")) And HarvestIsSufficient) Then
    Return 0
  ElseIf ((SolutionCostEstimate("Buy") > SolutionCostEstimate("Depot")) And DepotIsSufficient) Then
    Return 0
  ElseIf ((SolutionCostEstimate("Buy") > SolutionCostEstimate("Depot") + SolutionCostEstimate("Harvest")) And_
  HarvestDepotIsSufficient) Then
    Return 0
  Else : Return 1
  End If
End Function

Function IsHarvestUsed() As Integer
```

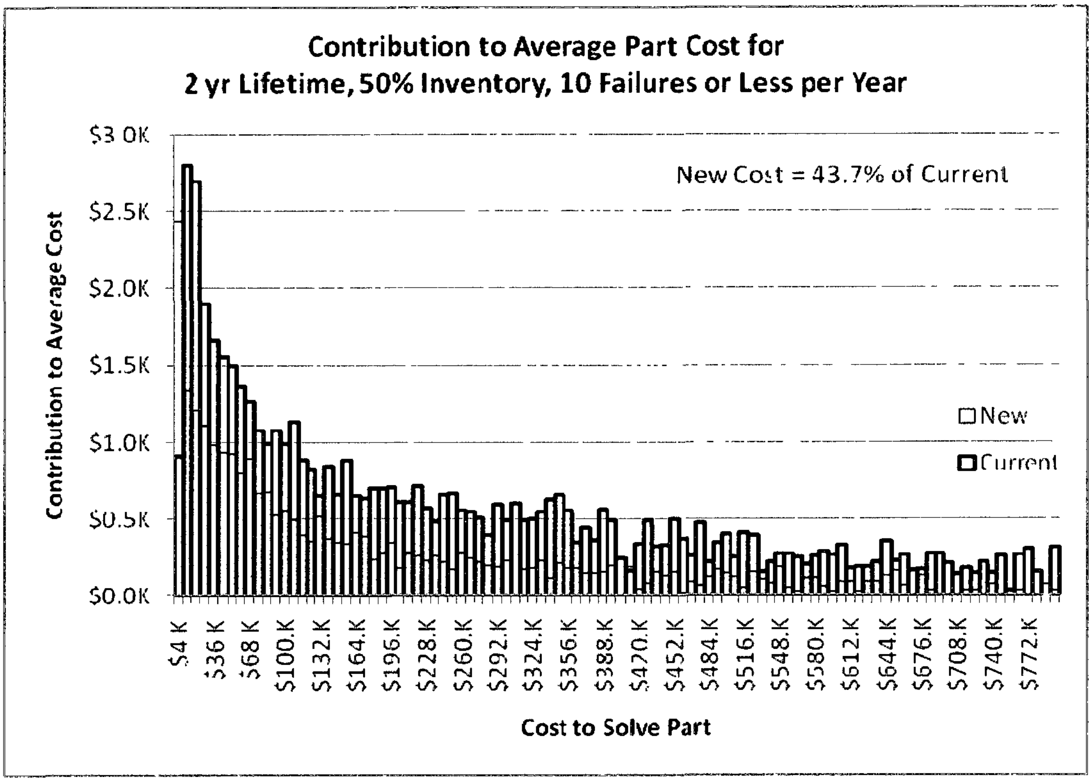
```
    If (Not HarvestOption) Then
        Return 0
    ElseIf (SolutionCostEstimate("Harvest") > SolutionCostEstimate("Buy")) Or (SolutionCostEstimate("Harvest") >_
    SolutionCostEstimate("Redesign")) Or (SolutionCostEstimate("Harvest") > SolutionCostEstimate("Substitution")) Then
        Return 0
    ElseIf ((SolutionCostEstimate("Harvest") > SolutionCostEstimate("Substitution")) And SubstitutionIsSufficient) Then
        Return 0
    ElseIf ((SolutionCostEstimate("Harvest") > SolutionCostEstimate("Depot")) And DepotIsSufficient) Then
        Return 0
    ElseIf StockIsSufficient Then
        Return 2
    Else : Return 1
    End If
End Function

Function IsSSBUsed() As Integer
    If (Not SSBOption) Then
        Return 0
    ElseIf StockIsSufficient Then
        Return 0
    ElseIf AvgSolutionOrder(0) = "SSB" Then
        Return 1
    ElseIf ((AvgSolutionOrder(0) = "Harvest") And (AvgSolutionOrder(1) = "SSB") And (Not HarvestIsSufficient)) Then
        Return 1
    ElseIf ((AvgSolutionOrder(0) = "Depot") And (AvgSolutionOrder(1) = "SSB") And (Not DepotIsSufficient)) Then
        Return 1
    ElseIf ((SolutionCostEstimate("SSB") < SolutionCostEstimate("Redesign")) And (SolutionCostEstimate("SSB") <_
    SolutionCostEstimate("Buy")) And (SolutionCostEstimate("SSB") < SolutionCostEstimate("Substitution")) And_
    (SolutionCostEstimate("SSB") < (SolutionCostEstimate("Harvest") + SolutionCostEstimate("Depot"))) And (Not_
   HarvestIsSufficient) And (Not DepotIsSufficient)) Then
      Return 1
   ElseIf ((SolutionCostEstimate("SSB") < SolutionCostEstimate("Redesign")) And (SolutionCostEstimate("SSB") <_
   SolutionCostEstimate("Buy")) And (SolutionCostEstimate("SSB") < SolutionCostEstimate("Substitution")) And (Not_
   HarvestDepotIsSufficient)) Then
      Return 1
   Else : Return 0
   End If
End Function

Function IsSubstitutionUsed() As Integer
   If (Not SubstitutionOption) Then
      Return 0
   ElseIf AvgSolutionOrder(0) = "Substitution" Then
      Return 1
   ElseIf ((AvgSolutionOrder(0) = "Harvest") And (AvgSolutionOrder(1) = "Substitution") And (Not HarvestIsSufficient)) Then
      Return 1
   ElseIf ((AvgSolutionOrder(0) = "Depot") And (AvgSolutionOrder(1) = "Substitution") And (Not DepotIsSufficient)) Then
      Return 1
   ElseIf ((SolutionCostEstimate("Substitution") < SolutionCostEstimate("Redesign")) And (SolutionCostEstimate("Substitution") <_
   SolutionCostEstimate("Buy")) And (SolutionCostEstimate("Substitution") < SolutionCostEstimate("SSB")) And_
   (SolutionCostEstimate("Substitution") < (SolutionCostEstimate("Harvest") + SolutionCostEstimate("Depot"))) And (Not_
   HarvestIsSufficient) And (Not DepotIsSufficient)) Then
      Return 1
   ElseIf ((SolutionCostEstimate("Substitution") < SolutionCostEstimate("Redesign")) And (SolutionCostEstimate("Substitution") <_
   SolutionCostEstimate("Buy")) And (SolutionCostEstimate("Substitution") < SolutionCostEstimate("SSB")) And (Not_
   HarvestDepotIsSufficient)) Then
      Return 1
   ElseIf SolutionCostEstimate("Substitution") < SolutionCostEstimate("Redesign") Then
```

```
      Return 2
   Else : Return 0
   End If
End Function

Function IsDepotUsed() As Integer
   If (Not DepotOption) Then
      Return 0
   ElseIf StockIsSufficient Then
      Return 0
   ElseIf (SolutionCostEstimate("Depot") > SolutionCostEstimate("Buy")) Or (SolutionCostEstimate("Depot") >_
   SolutionCostEstimate("Redesign")) Or (SolutionCostEstimate("Depot") > SolutionCostEstimate("SSB")) Then
      Return 0
   ElseIf ((SolutionCostEstimate("Depot") > SolutionCostEstimate("Substitution")) And SubstitutionIsSufficient) Then
      Return 0
   ElseIf ((SolutionCostEstimate("Depot") > SolutionCostEstimate("Harvest")) And HarvestIsSufficient) Then
      Return 0
   Else : Return 1
   End If
End Function

Function IsRedesignUsed() As Integer
   If (Not RedesignOption) Then
      Return 0
   ElseIf SolutionCostEstimate("Redesign") < SolutionCostEstimate("Substitution") Then
      Return 2
   Else : Return 0
   End If
End Function
```

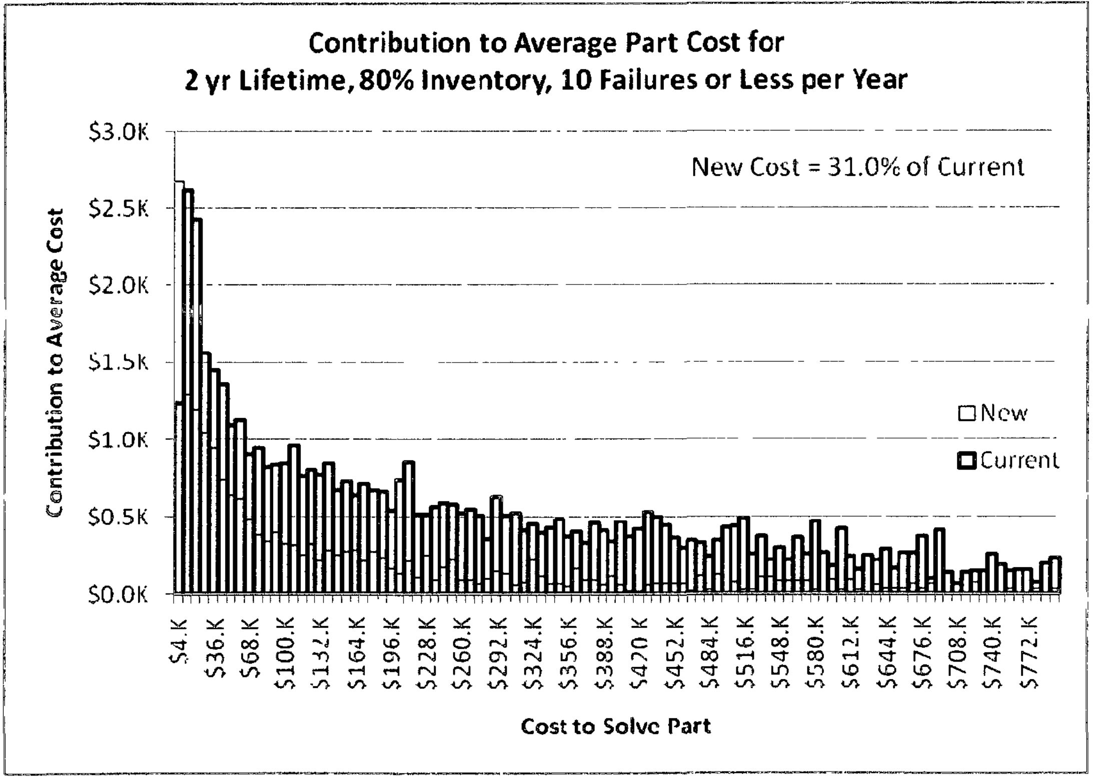
APPENDIX E: PERFORMANCE OF PROPOSED METHOD IN ADDITIONAL SITUATION



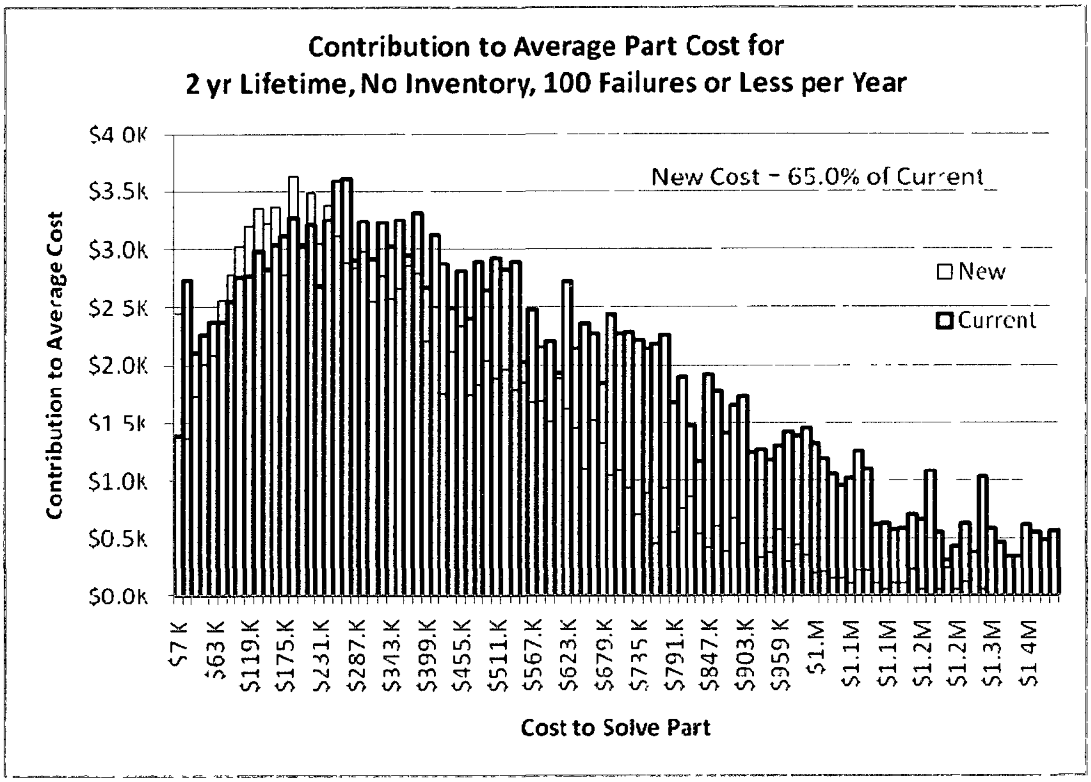
Appendix Graph 1



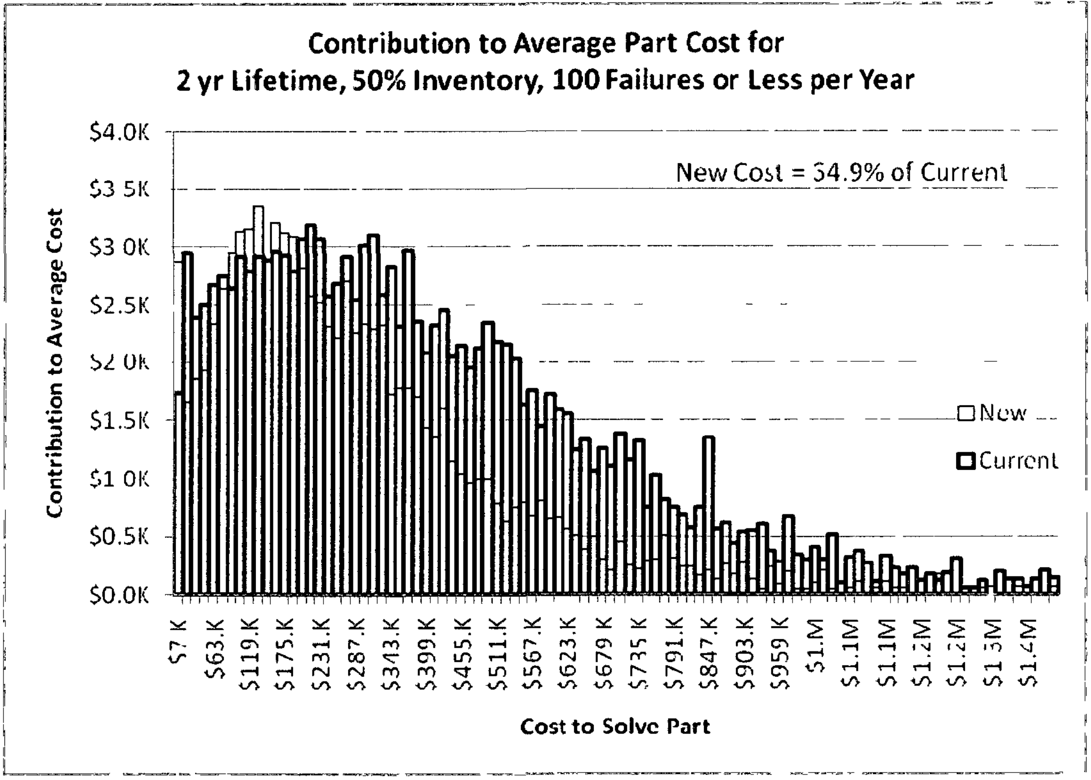
Appendix Graph 2



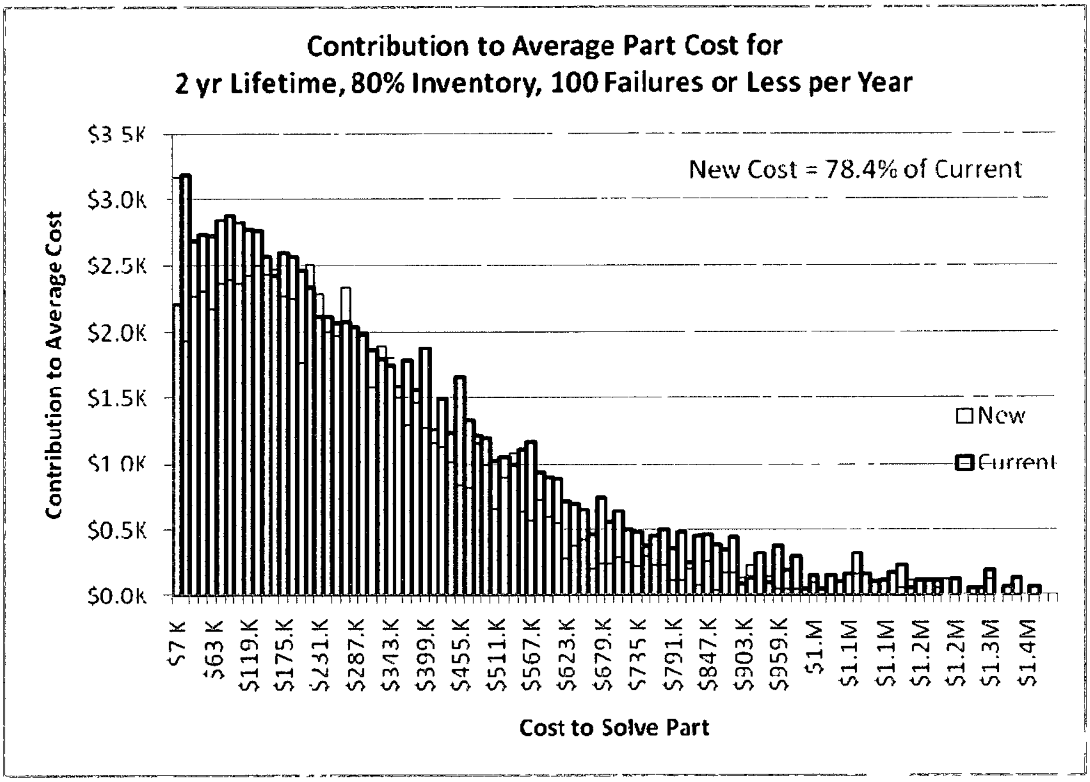
Appendix Graph 3



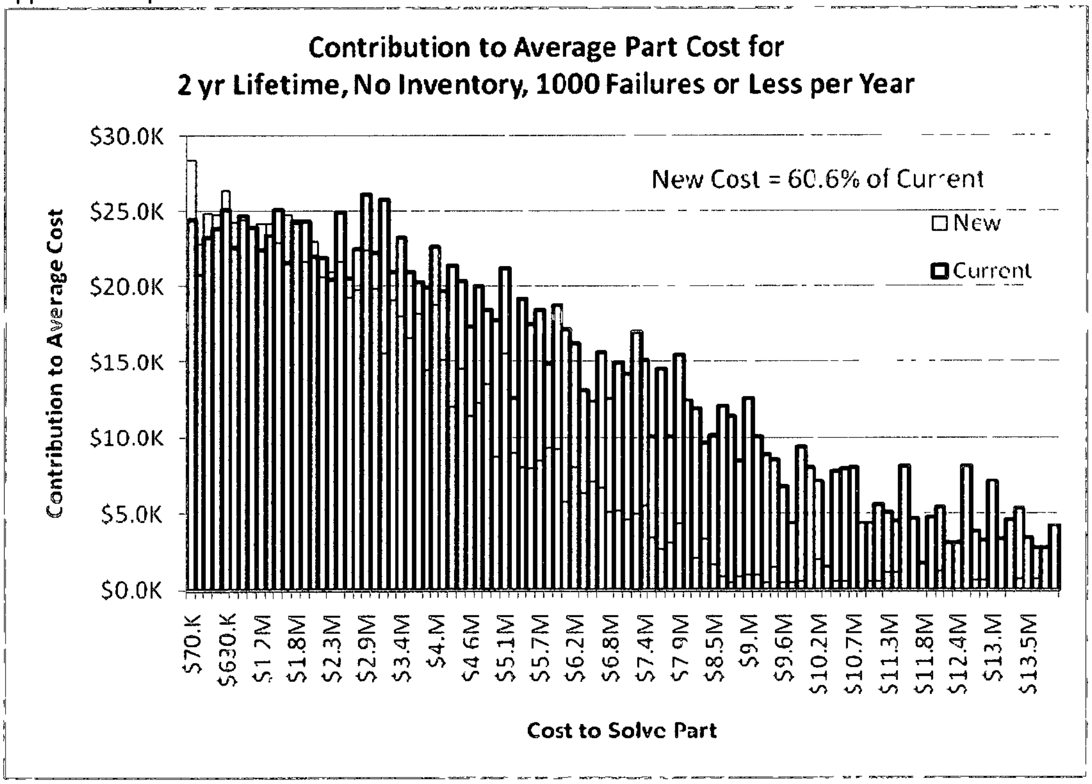
Appendix Graph 4



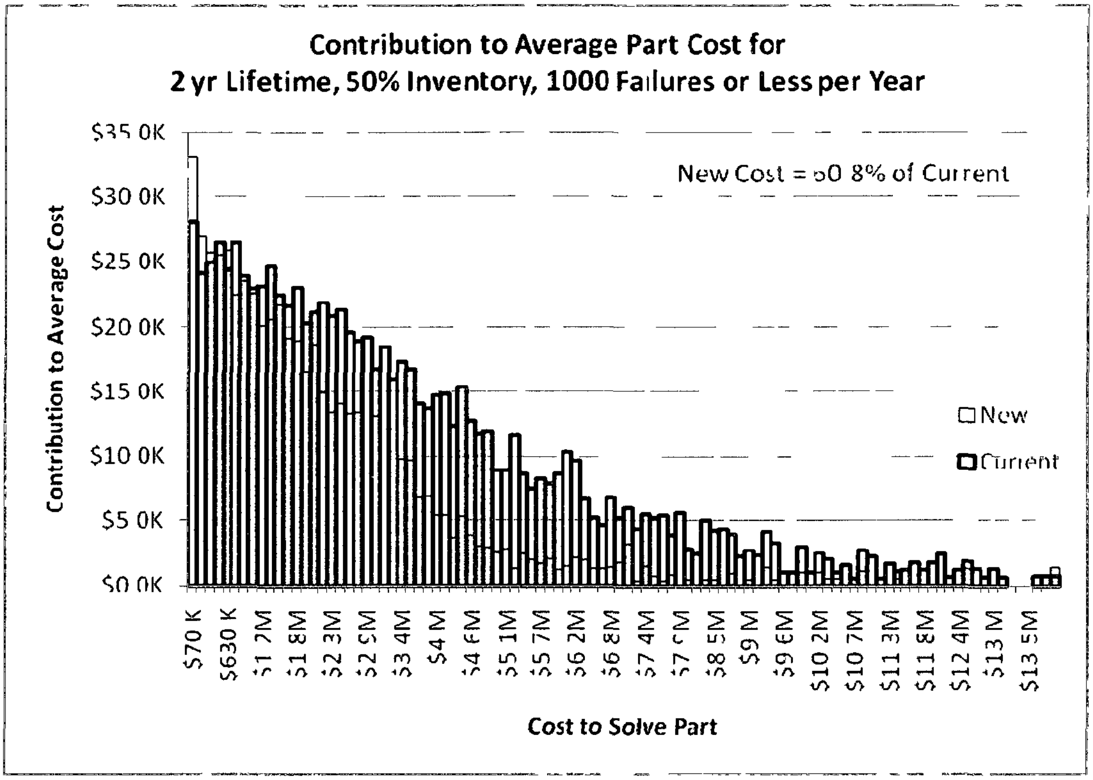
Appendix Graph 5



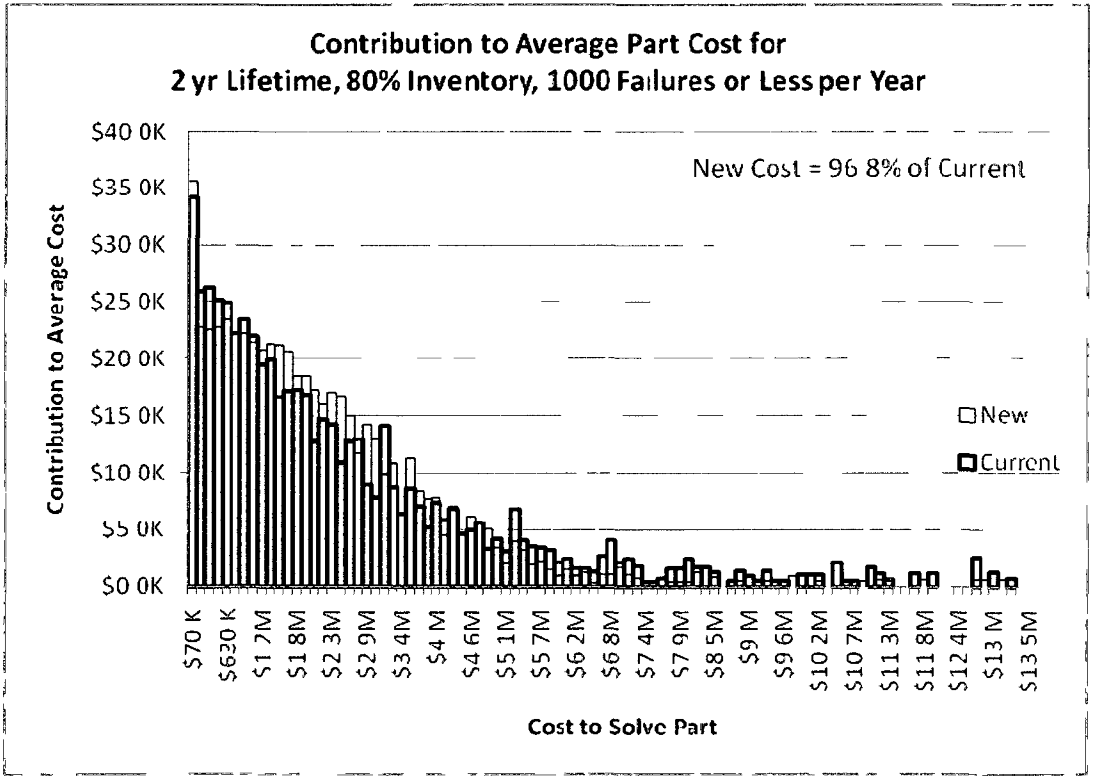
Appendix Graph 6



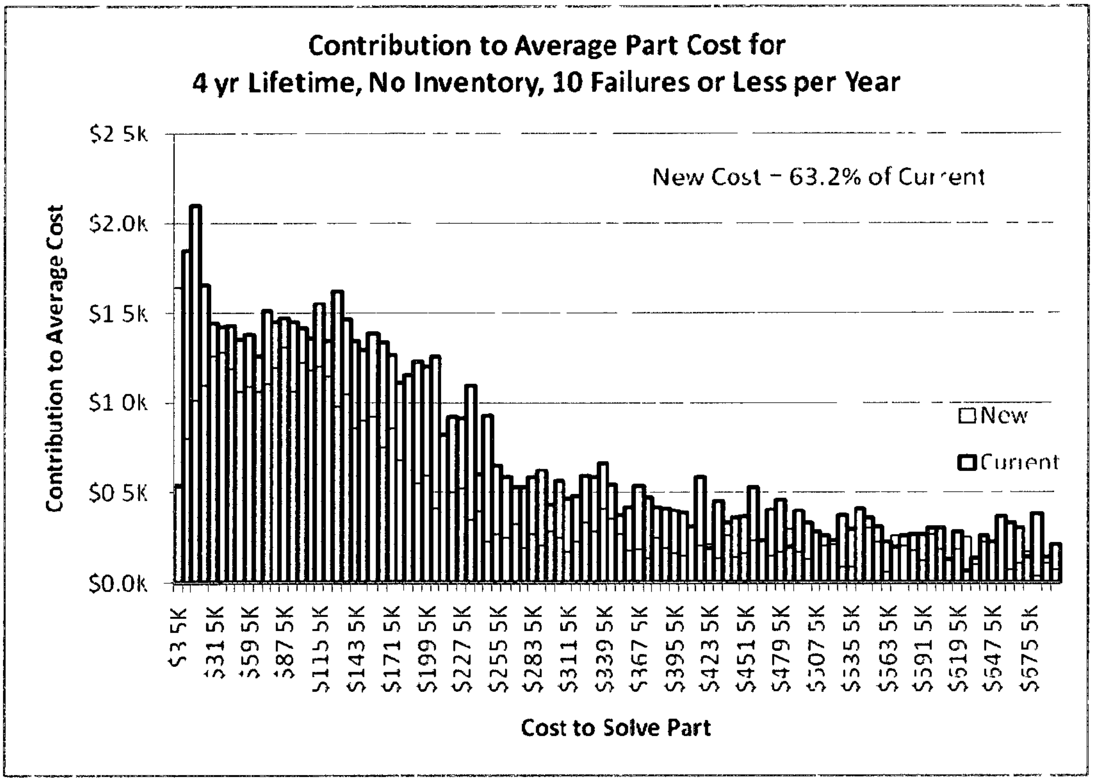
Appendix Graph 7



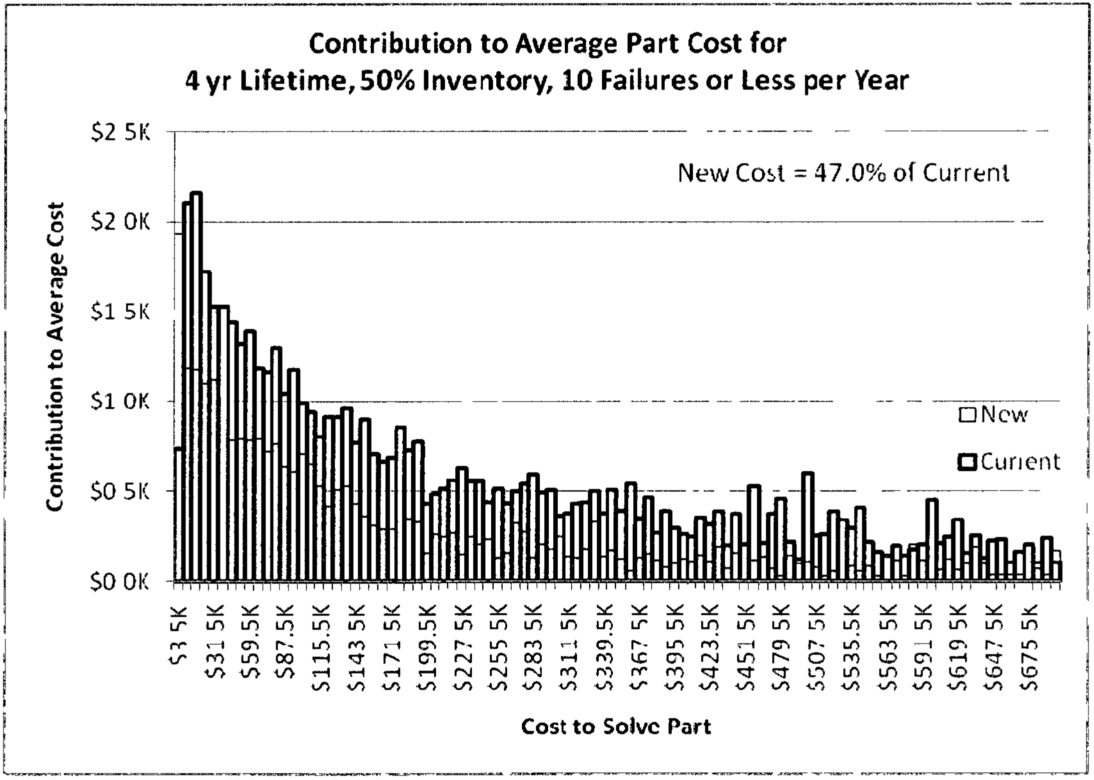
Appendix Graph 8



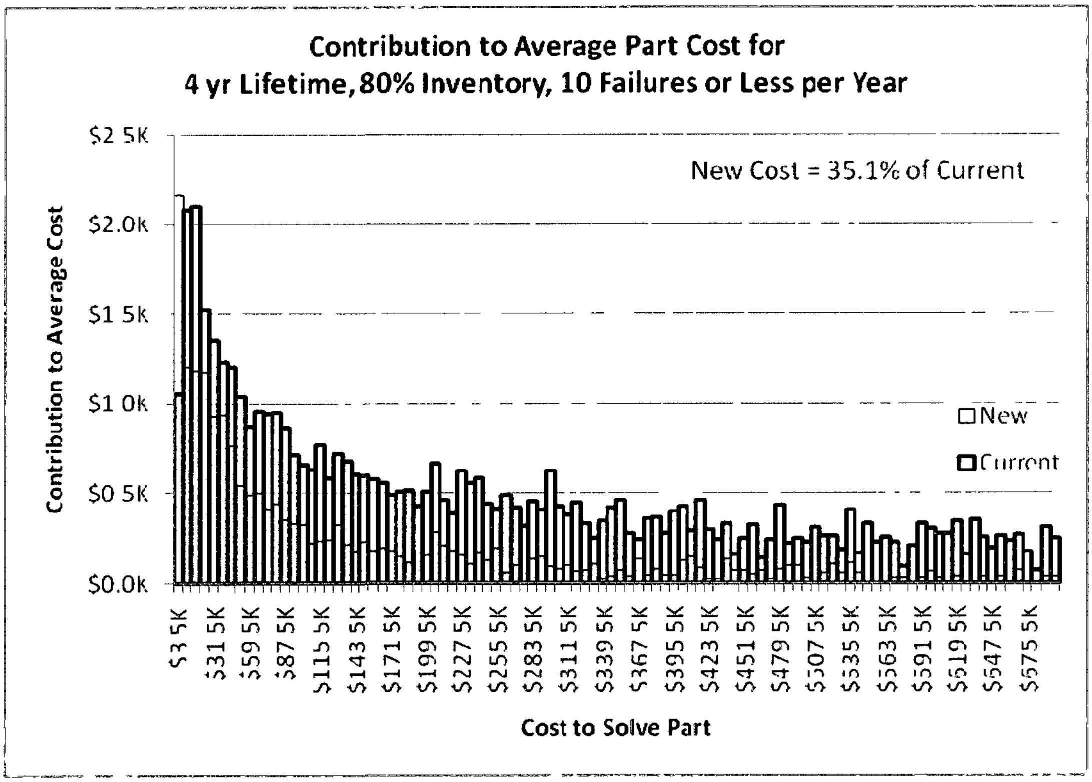
Appendix Graph 9



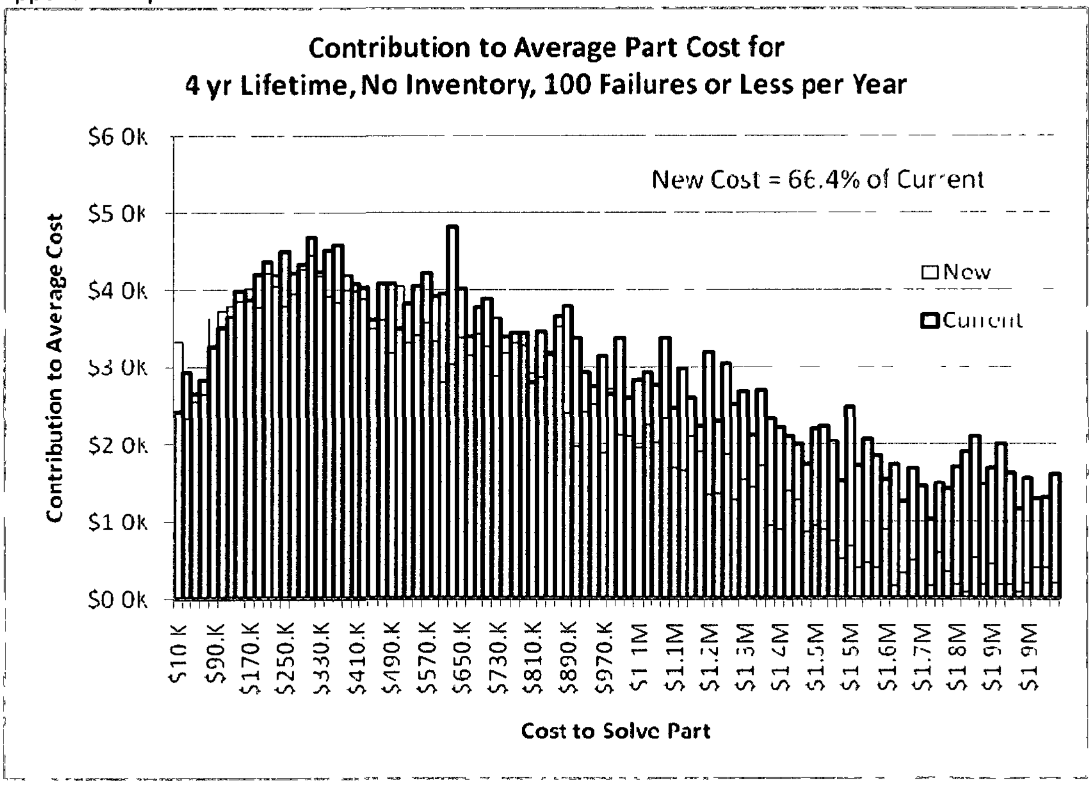
Appendix Graph 10



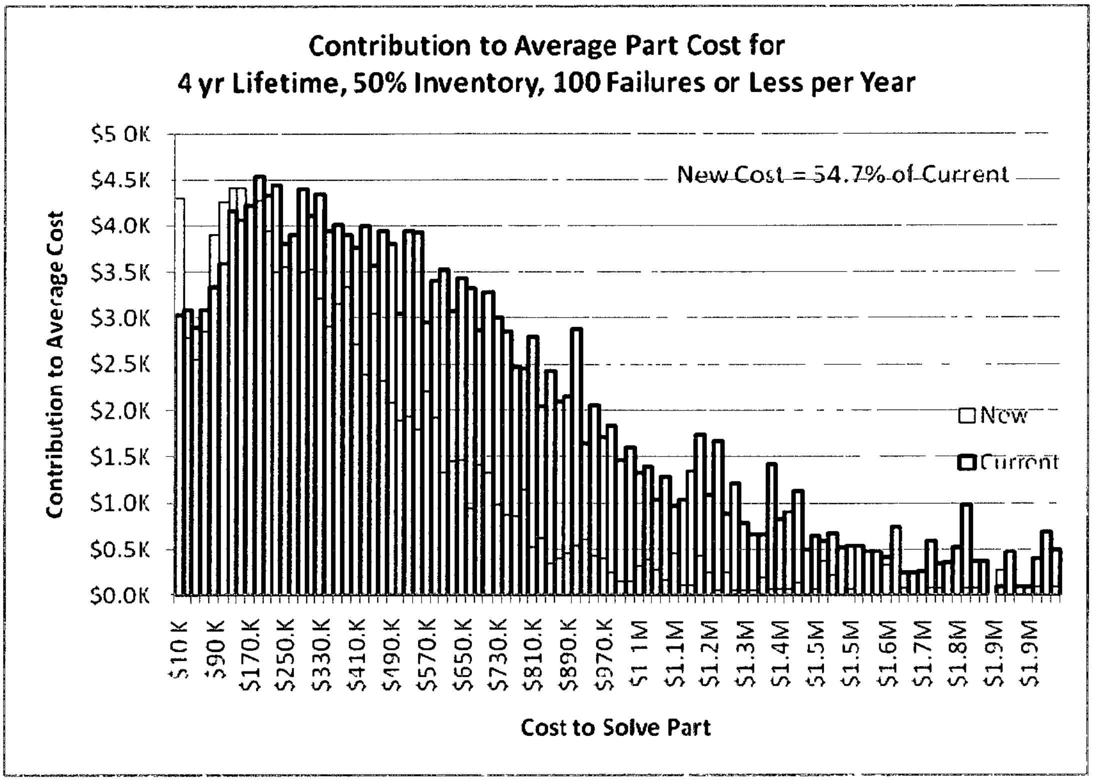
Appendix Graph 11


Contribution to Average Part Cost for
4 yr Lifetime, 80% Inventory, 10 Failures or Less per Year

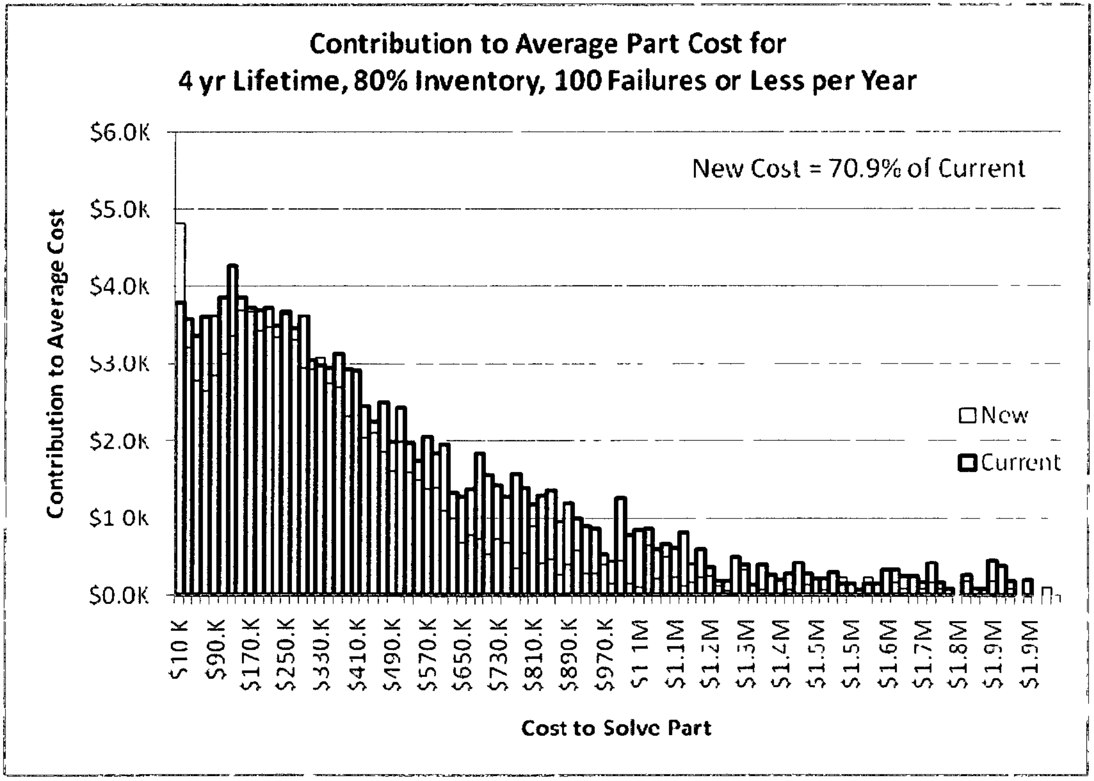
Appendix Graph 12


Contribution to Average Part Cost for
4 yr Lifetime, No Inventory, 100 Failures or Less per Year

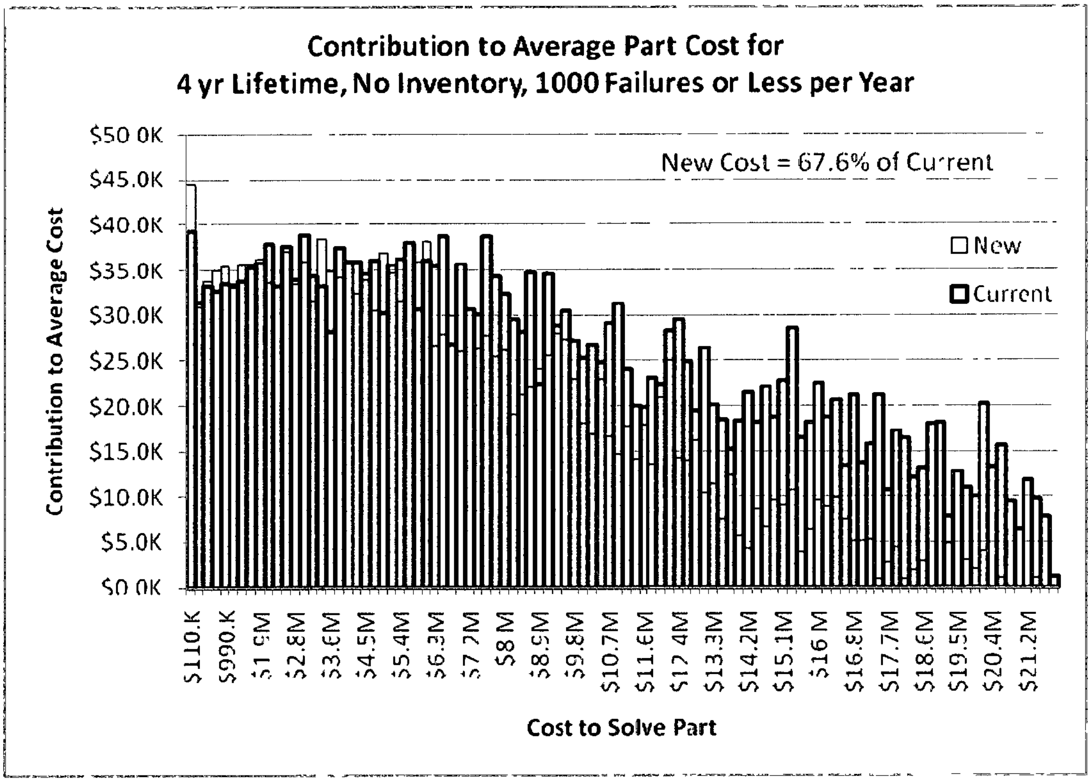
Appendix Graph 13



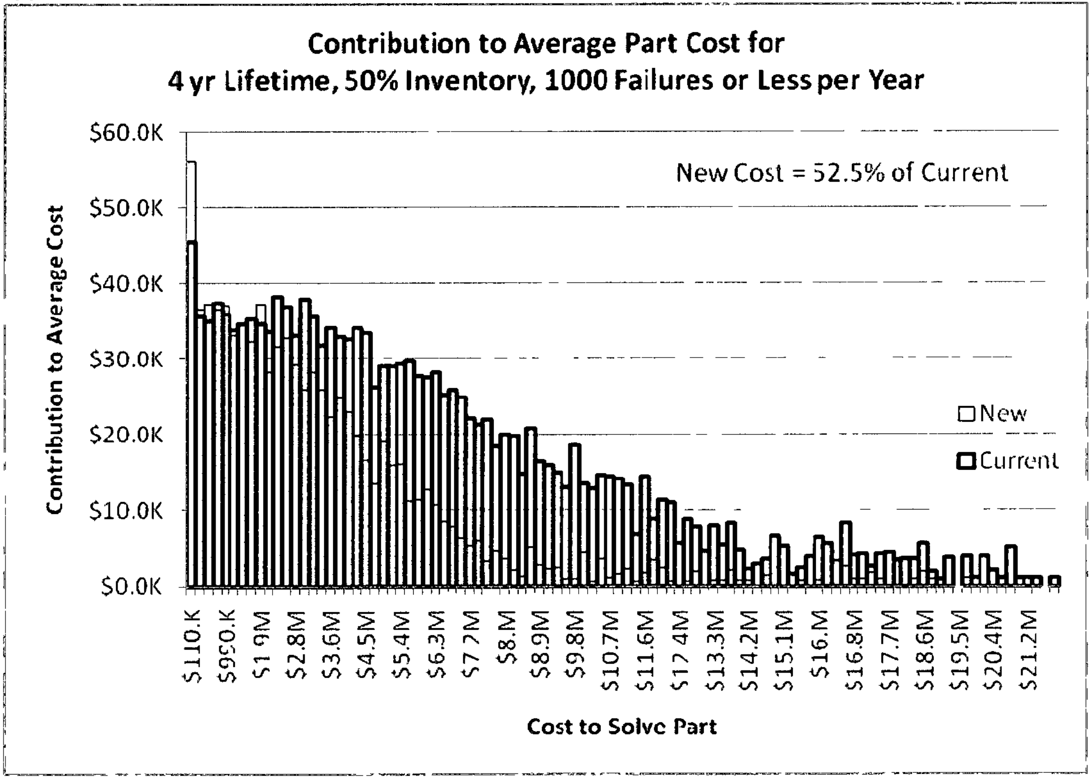
Appendix Graph 14



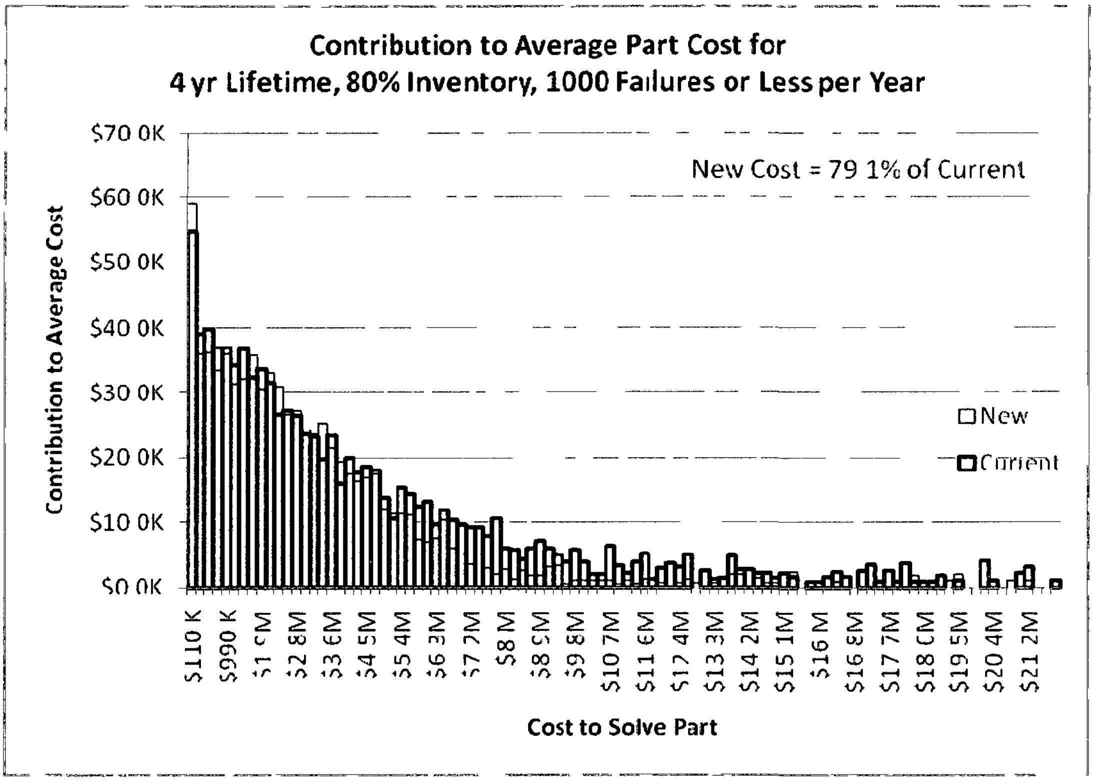
Appendix Graph 15



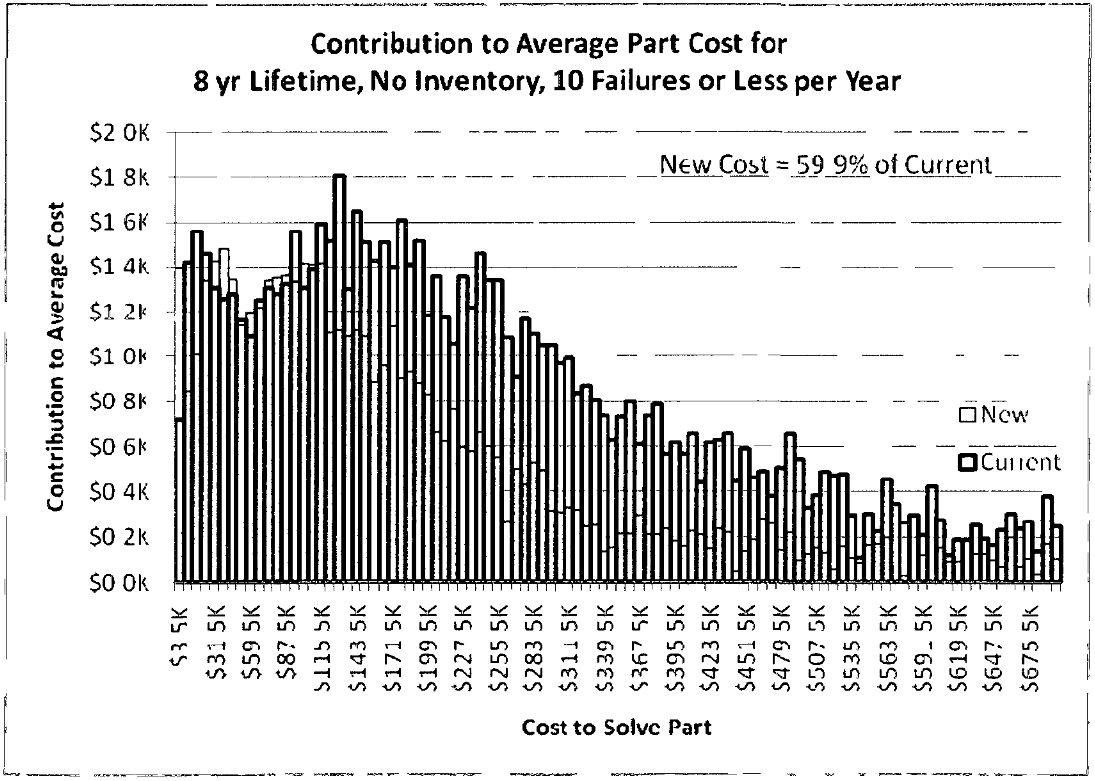
Appendix Graph 16



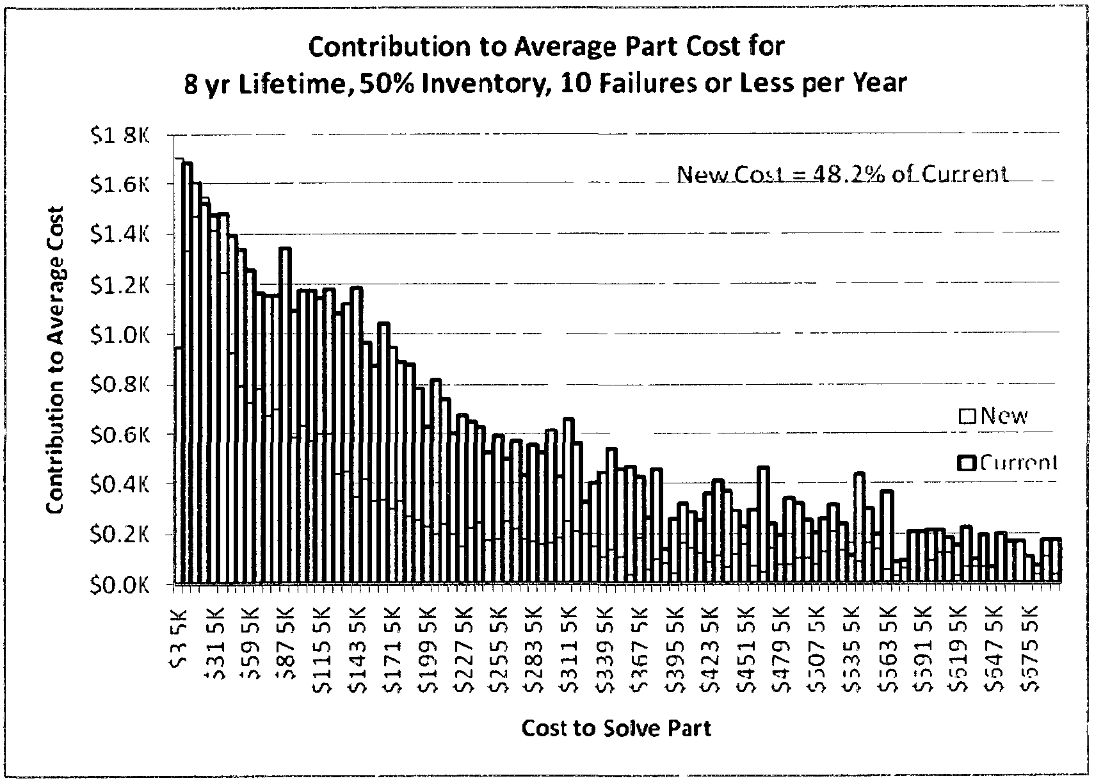
Appendix Graph 17



Appendix Graph 18

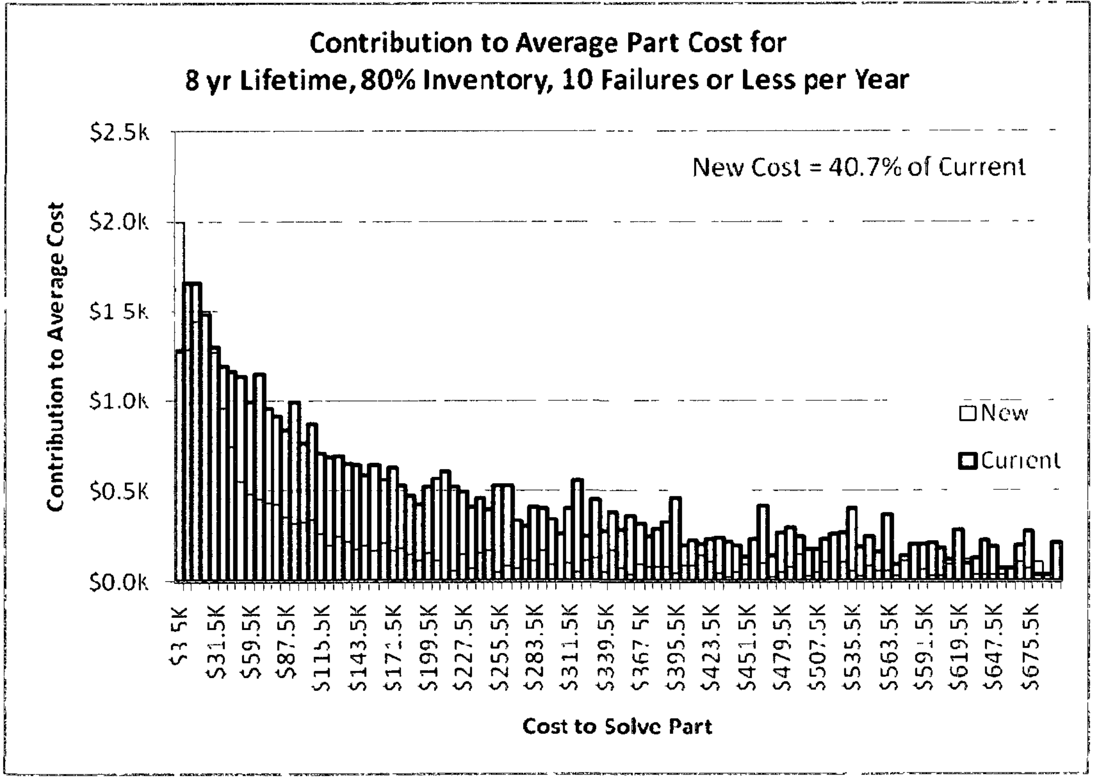


Appendix Graph 19



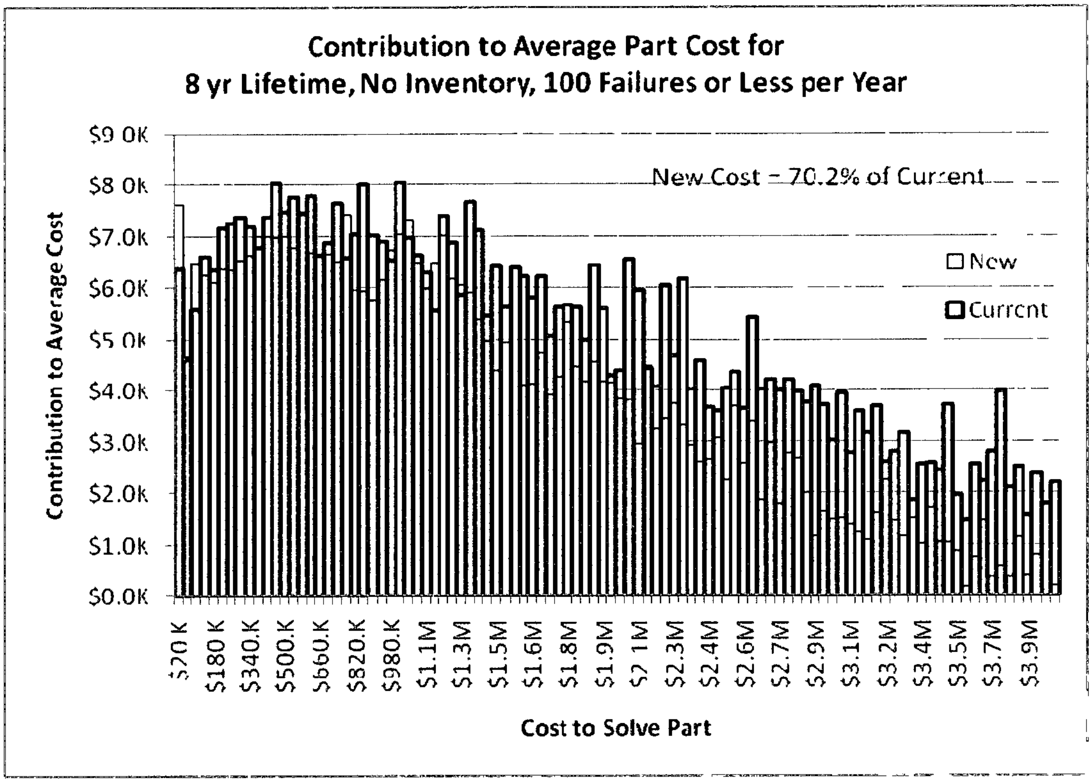
Contribution to Average Part Cost for
8 yr Lifetime, 50% Inventory, 10 Failures or Less per Year

Appendix Graph 20

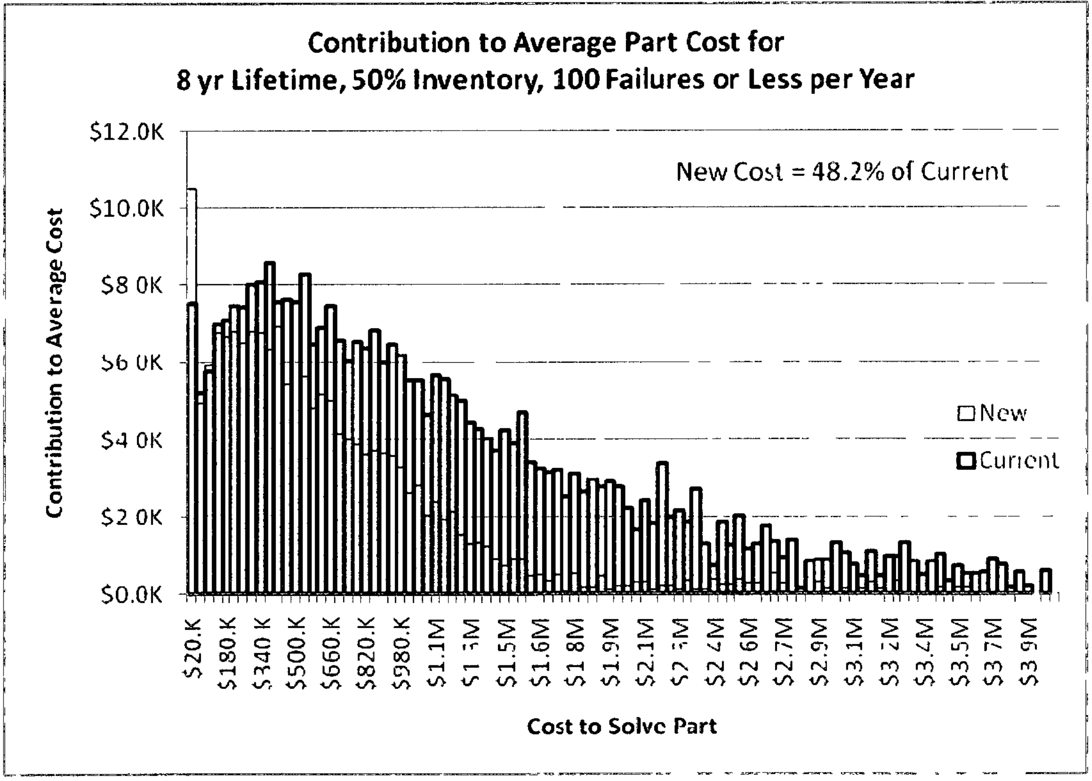


Contribution to Average Part Cost for
8 yr Lifetime, 80% Inventory, 10 Failures or Less per Year

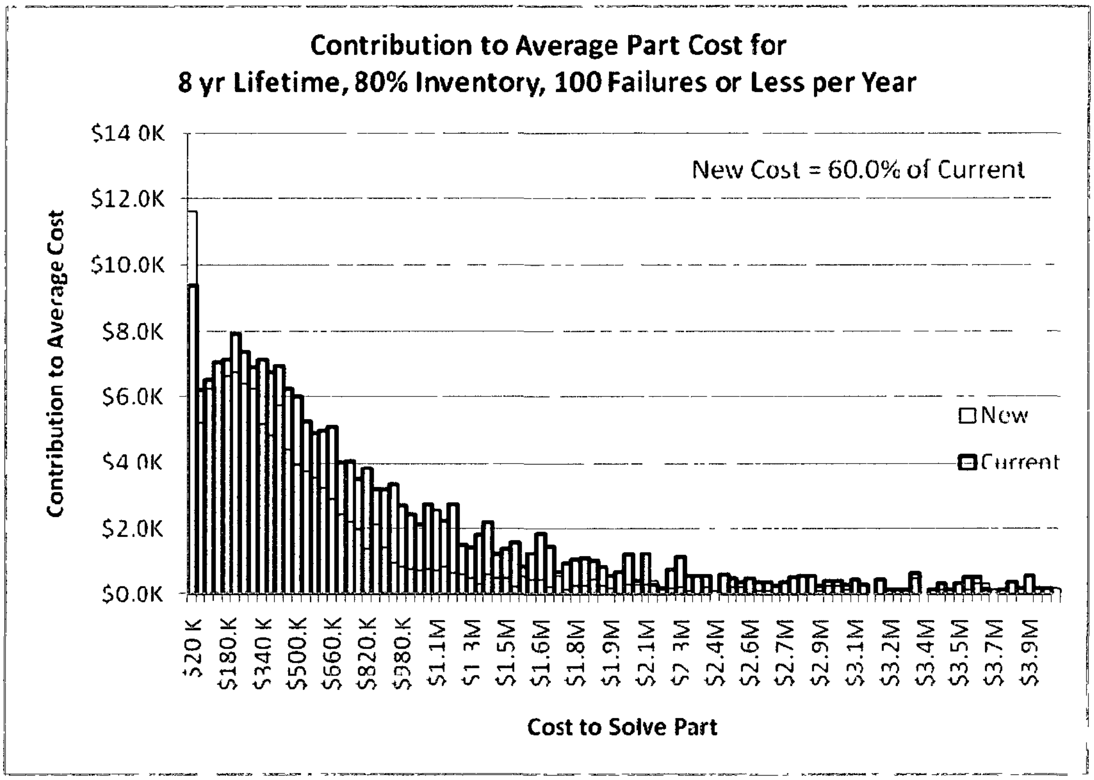
Appendix Graph 21



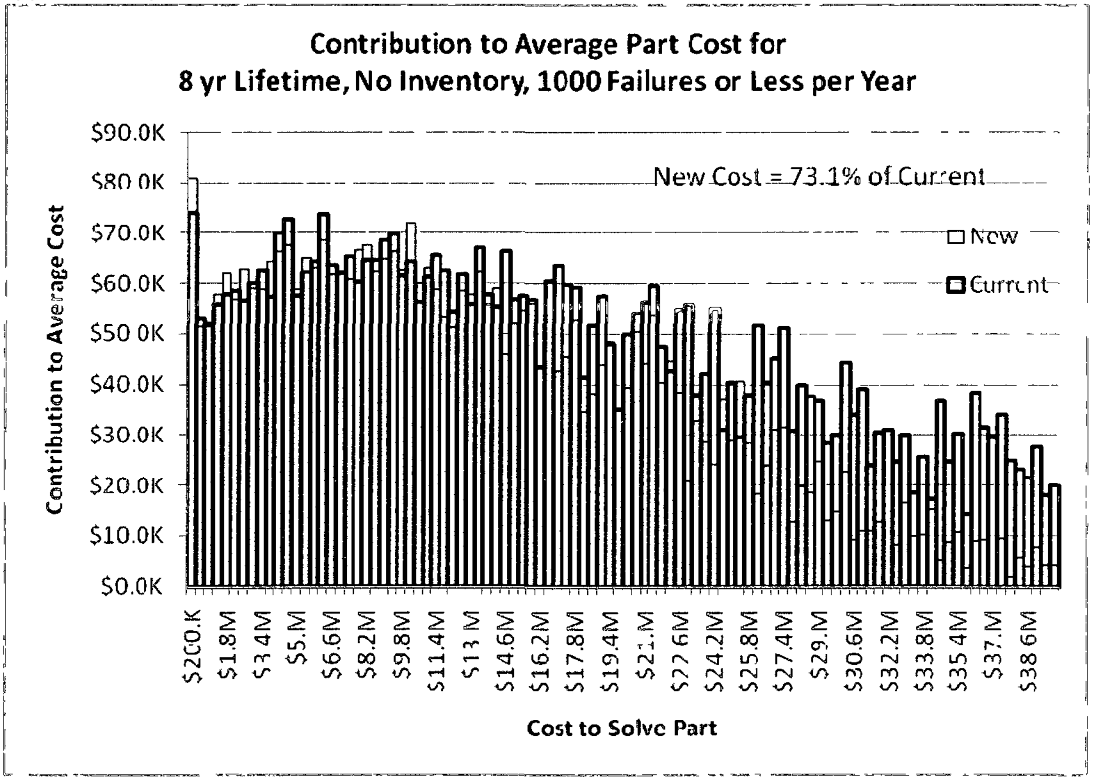
Appendix Graph 22



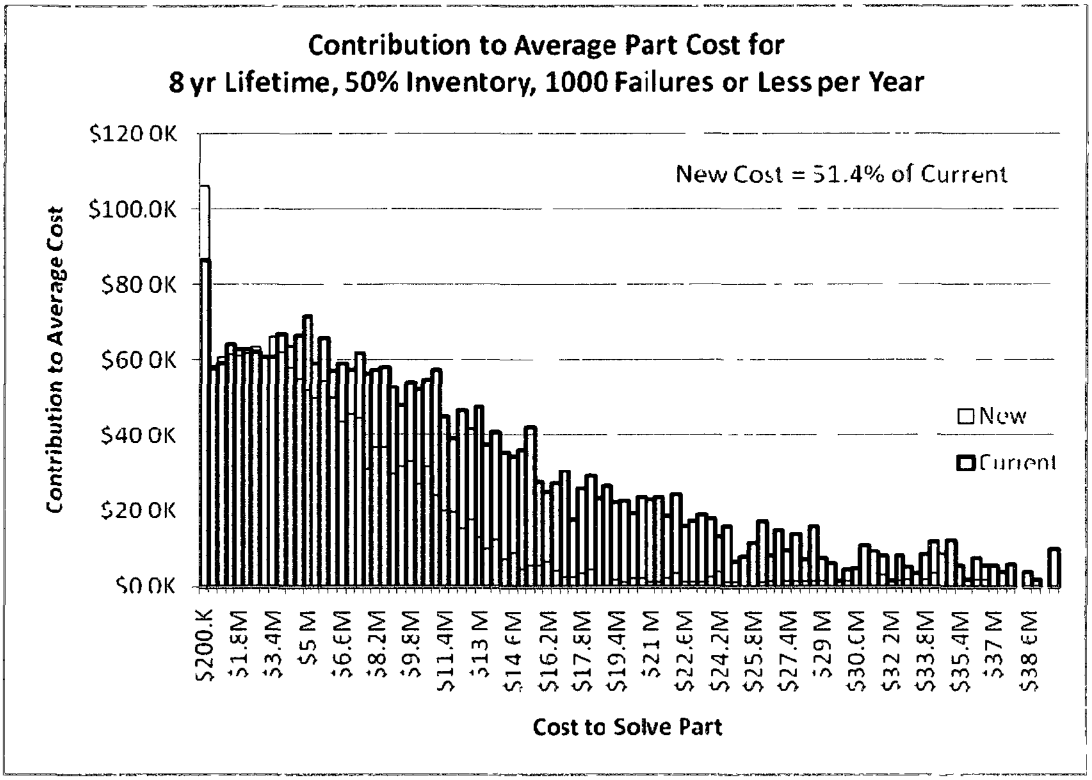
Appendix Graph 23



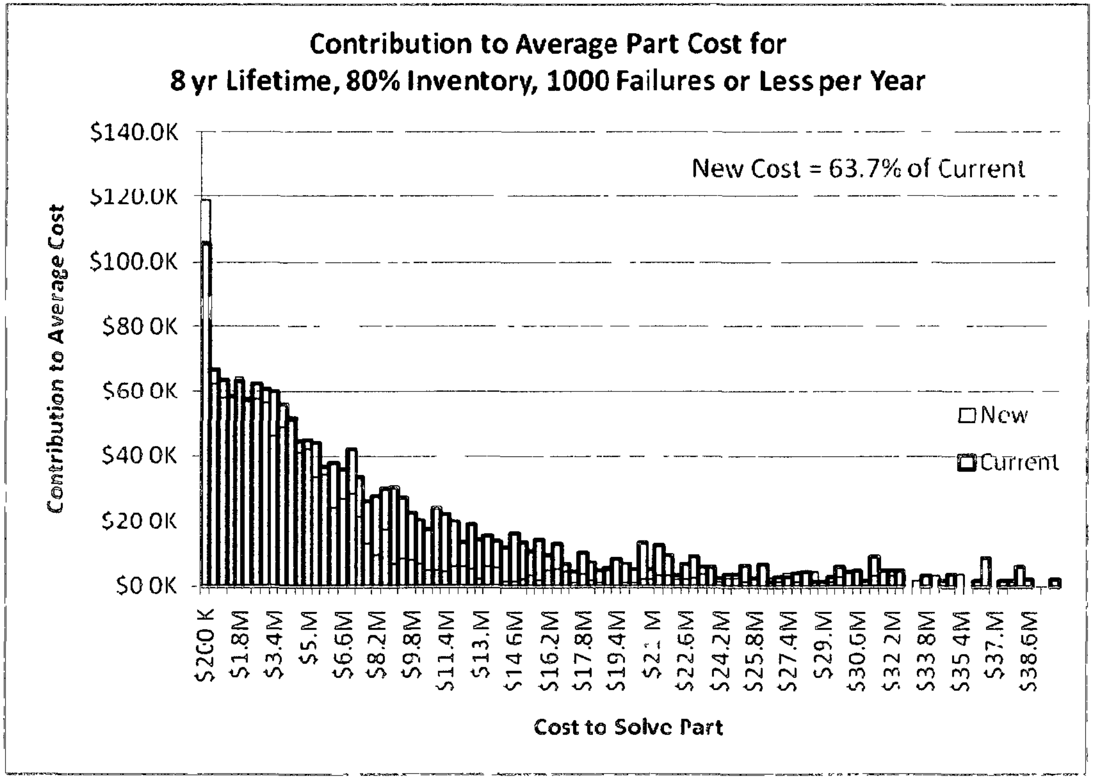
Appendix Graph 24



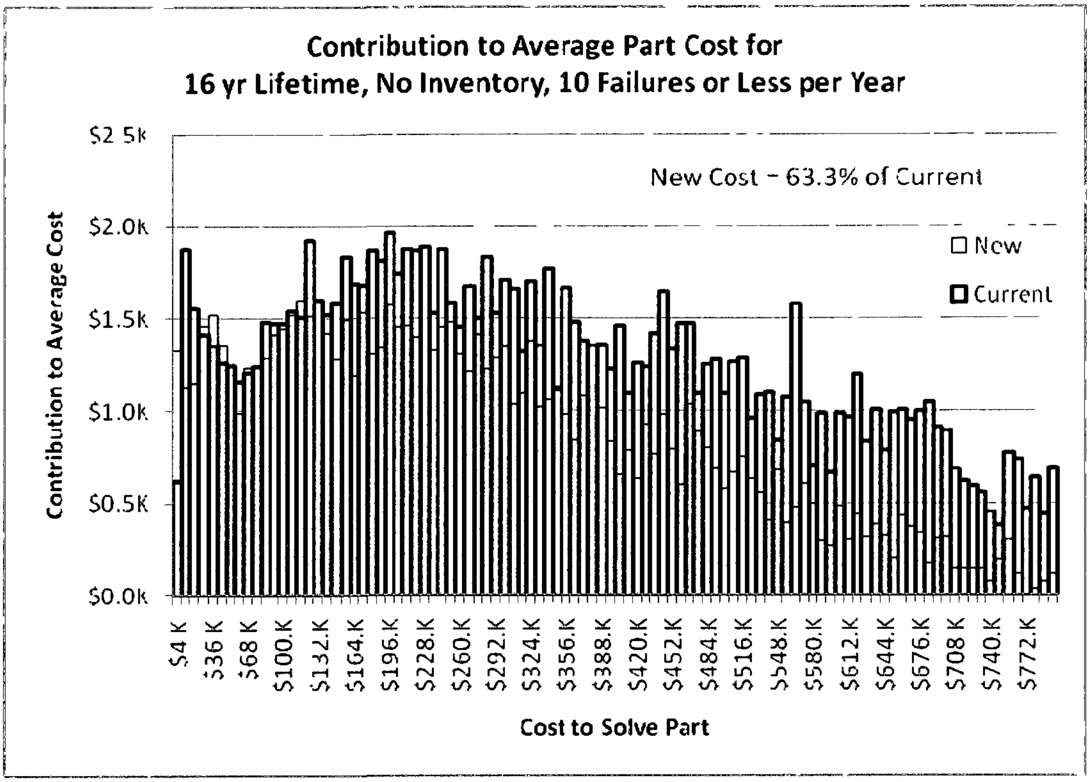
Appendix Graph 25



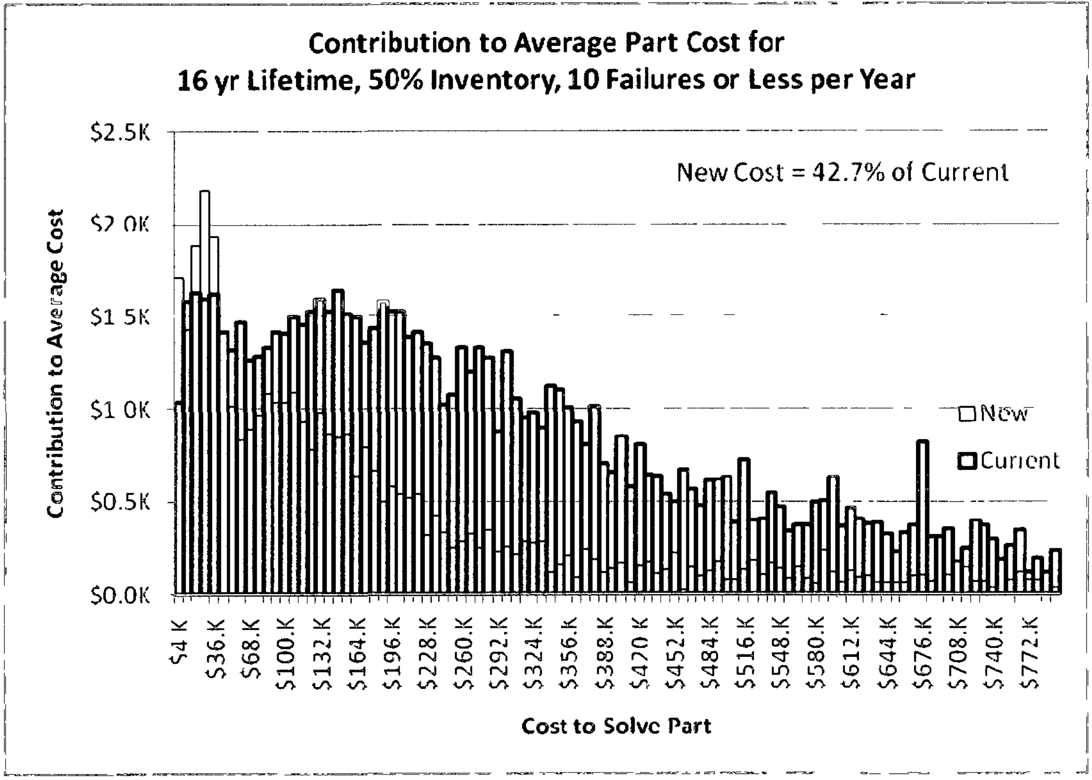
Appendix Graph 26



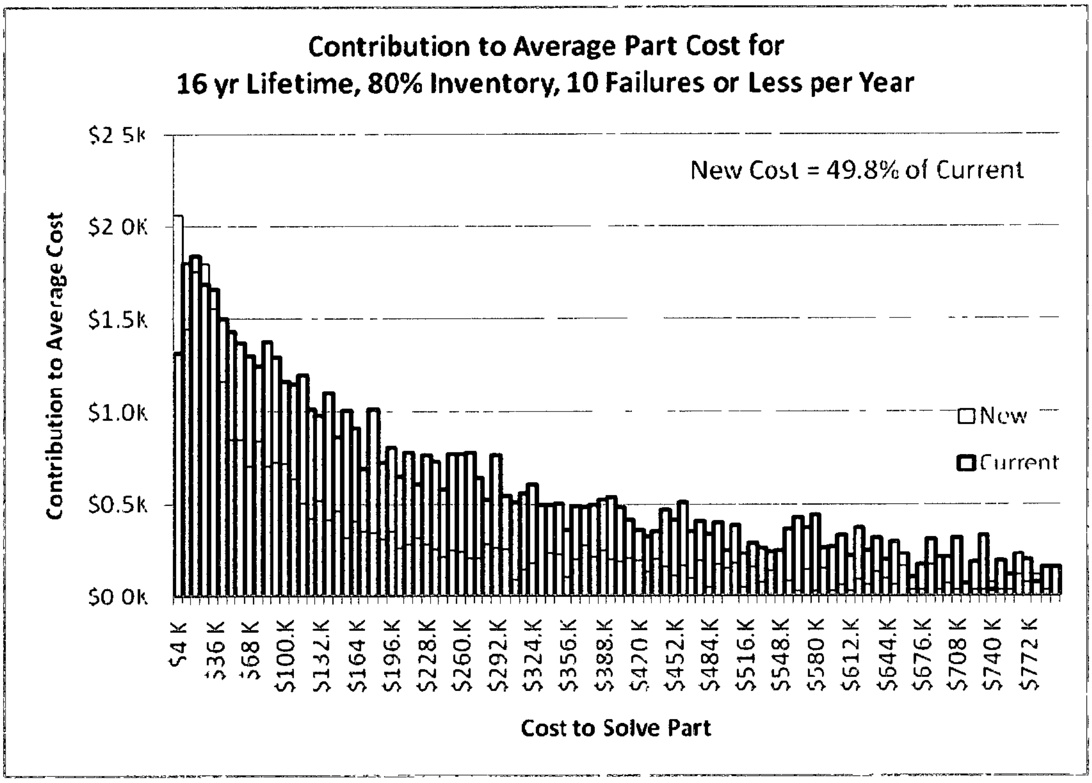
Appendix Graph 27



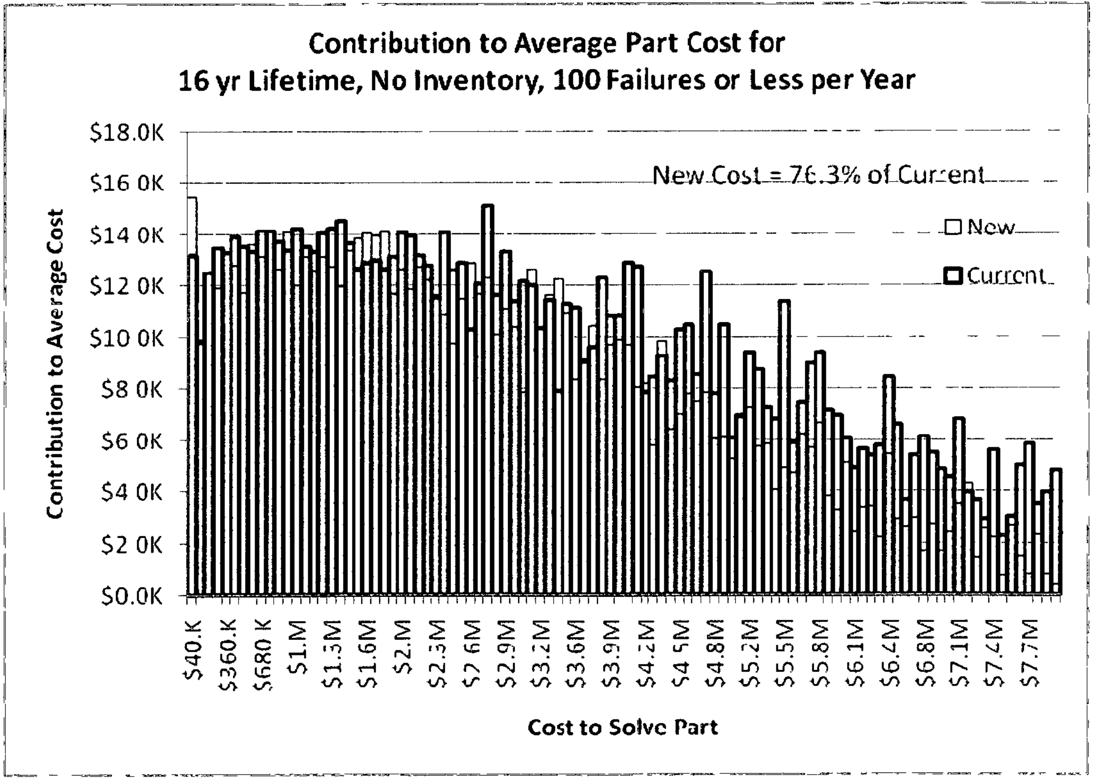
Appendix Graph 28



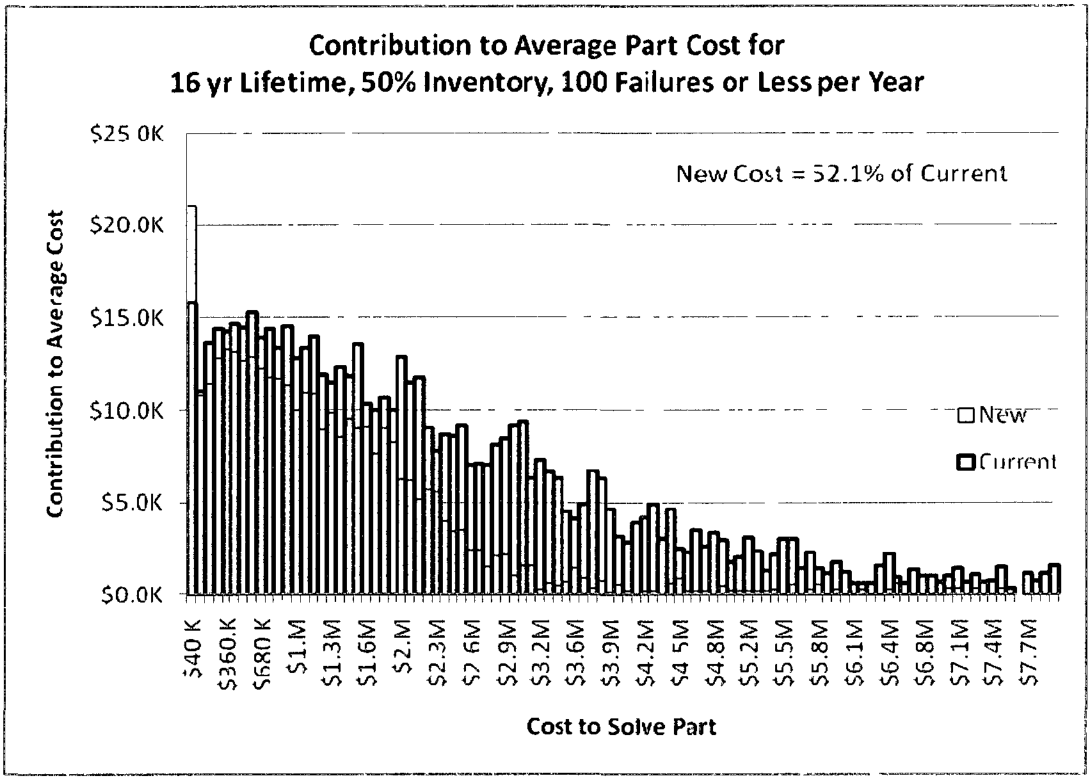
Appendix Graph 29



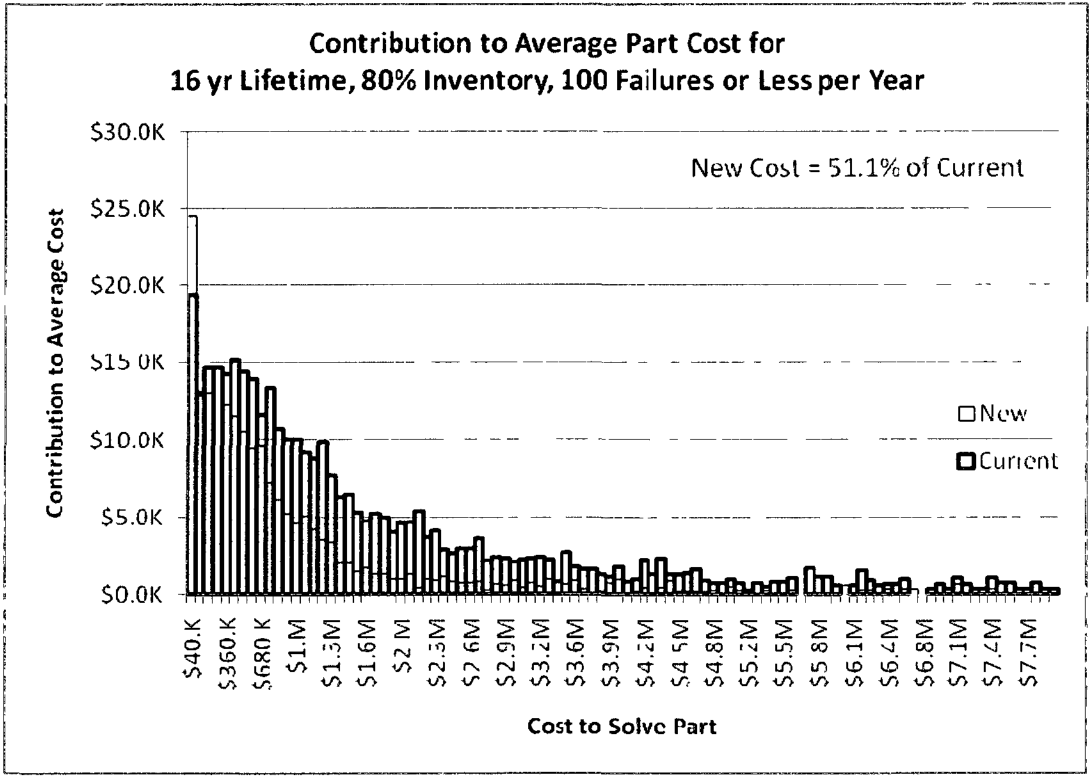
Appendix Graph 30



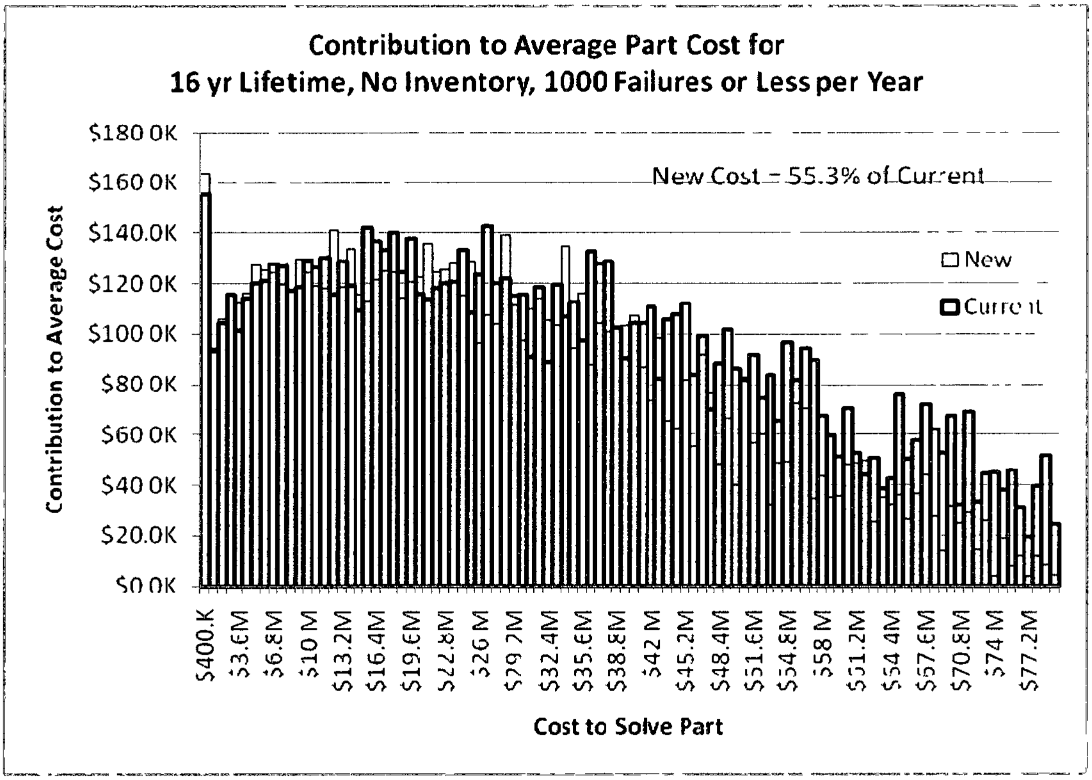
Appendix Graph 31



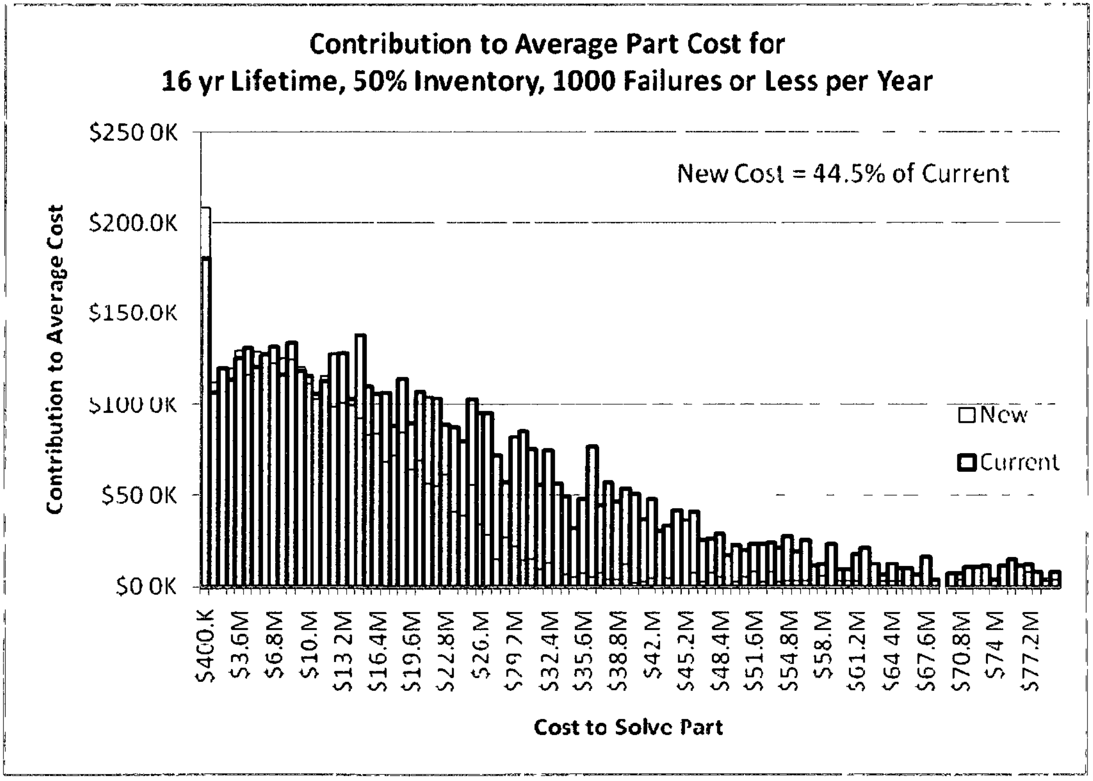
Appendix Graph 32



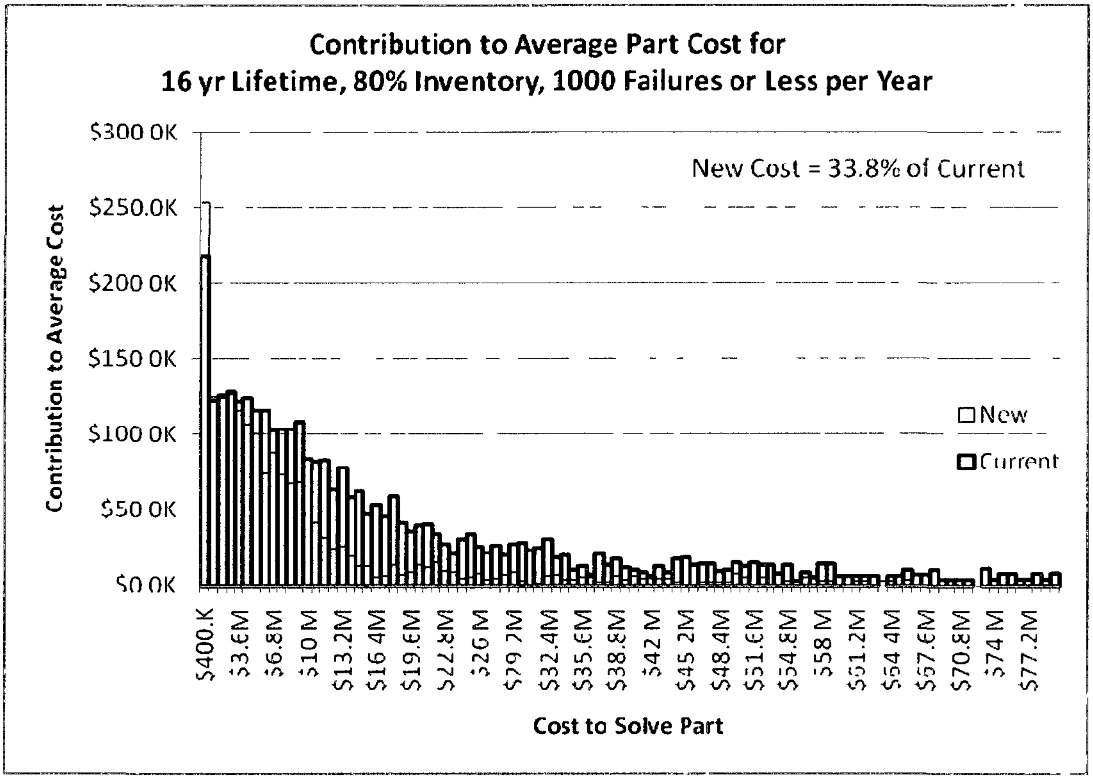
Appendix Graph 33



Appendix Graph 34



Appendix Graph 35



Appendix Graph 36

VITA – MICHAEL ASHTON GAINTNER

After graduating from Rose-Hulman Institute of Technology in 2004 with a Bachelor of Science in Electrical Engineering, Michael Gaintner began his professional career at the Naval Surface Warfare Center in Virginia Beach, VA. While working for the Navy, Dr. Gaintner obtained a Master of Science degree in Electrical Engineering from Virginia Tech and a Doctor of Philosophy in Engineering Management from Old Dominion University.